



*Программирование  
и администрирование  
СУБД*

Microsoft®  
**SQL Server**

# Урок №5

## Содержание

---

1. Директивы COMPUTE и COMPUTE BY .....3
2. Надагрегатные операторы ROLLUP,  
CUBE и GROUPING SETS .....7
3. Операторы PIVOT и UNPIVOT .....18
4. Представления в MS SQL Server.....23
5. Домашнее задание .....35

# 1. Директивы COMPUTE и COMPUTE BY

Операторы COMPUTE и COMPUTE BY создают новые строки на основе данных, которые возвращаются оператором SELECT. В них используются функции агрегирования.

Обобщенный синтаксис использования:

```
SELECT запрос
[ COMPUTE
    { { AVG | COUNT | MAX | MIN | STDEV | STDEVP | VAR |
      VARP | SUM } ( выражение ) } [, ...n]
    [ BY выражение [ , ...n ] ]
]
```

Оператор **COMPUTE** генерирует результирующие значения, которые отображаются в виде дополнительных строк.

Оператор **COMPUTE BY** возвращает новые строки для групповых данных, что похоже на директиву GROUP BY, но здесь строки возвращаются как подгруппы с рассчитанными значениями.

Кстати, в одном запросе можно одновременно указать оператор COMPUTE и COMPUTE BY, но в следующей версии MS SQL Server эту возможность планируется устранить.

Например, напишем запрос, который выводит на экран общую стоимость книг каждой тематики. Используем при написании данного запроса привычный нам оператор GROUP BY, ведь без него при построении данного запроса никак не обойтись.

```
select t.NameTheme, sum(b.Price)
from book.Books b, book.Themes t
where b.id_theme = t.id_theme
group by t.NameTheme;
```

Результат:

Results Messages		
	NameTheme	(No column name)
1	Computer Science	891,93
2	Science Fiction	137
3	Web Technologies	127

При использовании оператора COMPUTE мы получим дополнительное поле с итоговой суммой цен всех книг:

```
select t.NameTheme, b.Price
from book.Books b, book.Themes t
where b.id_theme = t.id_theme
-- order by t.NameTheme
-- можно включить сортировку по произвольному полю
compute sum(b.Price);
```

Результат:

Results		Messages
	Name Theme	price
1	Science Fiction	25
2	Science Fiction	27
3	Science Fiction	22
4	Science Fiction	25
5	Computer Science	43
6	Computer Science	23
7	Computer Science	69
8	Computer Science	45
	sum	
1	1155,93	

Итоговая сумма

При использовании оператора **COMPUTE BY** в запрос **ОБЯЗАТЕЛЬНО** включается директива **ORDER BY**. При этом, если **ORDER BY** имеет вид:

```
ORDER BY t.NameTheme, b.NameBook
    то COMPUTE BY должен быть одним из следующих вариантов
-- 1 вариант
COMPUTE функция_агрегирования (поле)
BY t.NameTheme, b.NameBook
-- 2 вариант
COMPUTE функция_агрегирования (поле)
BY t.NameTheme
```

То есть поля, которые перечислены в операторе **COMPUTE BY** те же, что и в **ORDER BY**, или составляют их подмножество. Последовательность полей в **COMPUTE BY** должна быть такой же, как и в **ORDER BY**, пропускать поля нельзя.

Итак, перепишем наш запрос с использованием оператора COMPUTE BY:

```
select t.NameTheme, b.Price
from book.Books b, book.Themes t
where b.id_theme = t.id_theme
order by t.NameTheme
compute sum(b.Price)
by t.NameTheme;
```

Следует отметить, что при использовании данного оператора нельзя применять оператор SELECT INTO, поскольку COMPUTE и COMPUTE BY создают новые записи (строки) нереляционных данных. При использовании вышеописанных операторов также нельзя использовать данные типа text или image, поскольку они не подлежат упорядочиванию.

Результат:

Results		Messages
	NameTheme	price
7	Computer Science	15
	sum	
1	891,93	
	NameTheme	price
1	Science Fiction	25
	sum	
1	137	
	NameTheme	price
1	Web Technologies	45
	sum	
1	127	

Итоговая сумма  
по каждой тематике

## 2. Надаггрегатные операторы ROLLUP, CUBE и GROUPING SETS

Операторы ROLLUP, CUBE GROUPING SETS задекларированы стандартом ANSI / ISO SQL'99 и используются для создания дополнительных строк в результате выполнения команды SELECT. Данные операторы называют "надаггрегатными", поскольку для своей работы они используют функции агрегирования и используются вместе с оператором GROUP BY.

Обобщенный синтаксис использования выглядит следующим образом:

```
SELECT запрос
[GROUP BY
    [ { ALL | CUBE | ROLLUP | GROUPING SETS } ]
    выражение_группирования]
[WITH {CUBE | ROLLUP} ]
-- альтернативный вариант э WITH
```

Стоит отметить, что поддержку ANSI стандарта для операторов ROLLUP, CUBE и GROUPING SETS Microsoft реализовал только в версии SQL Server 2008 (синтаксис с "WITH" является устаревшим и не рекомендован к применению, но поддерживается для совместимости с ранними версиями).

Оператор **ROLLUP** чаще всего используют для расчета средних значений или сумм. Он задает агрегатную функцию для набора полей оператора SELECT с директивой GROUP BY, обрабатывая их слева направо.

Для лучшего понимания работы вышеупомянутого оператора рассмотрим сначала пример без использования агрегатов. Напишем запрос, в результате которого получим общую сумму книг каждой тематики и каждого автора.

```
select t.NameTheme, a.FirstName + ' ' + a.LastName as
'FullName', sum(b.Price)
from book.Books b, book.Themes t, book.Authors a
where b.id_theme = t.id_theme and b.id_author =
a.id_author
group by t.NameTheme, a.FirstName + ' ' + a.LastName
```

Результат:

	NameTheme	Full Name	(No column name)
1	Science Fiction	Clifford Simak	99
2	Web Technologies	Dino Esposito	81
3	Computer Science	Donald Knuth	142
4	Computer Science	Herbert Schildt	126
5	Science Fiction	Herbert Wells	38
6	Computer Science	Jeffrey Richter	135
7	Computer Science	Matthew MacDonald	135
8	Web Technologies	Matthew MacDonald	46
9	Computer Science	Richard Waymire	253,93

А теперь перепишем данный запрос с использованием оператора ROLLUP:

```
select t.NameTheme, a.FirstName + ' ' + a.LastName as
'FullName',
sum(b.Price) as 'Total price'
from book.Books b, book.Themes t, book.Authors a
where b.id_theme = t.id_theme and b.id_author =
a.id_author
group by t.NameTheme, a.FirstName + ' ' + a.LastName
with rollup;
```



```
-- ИЛИ
select t.NameTheme, a.FirstName + ' ' + a.LastName as
'FullName',
       sum(b.Price) as 'Total price'
from book.Books b, book.Themes t, book.Authors a
where b.id_theme = t.id_theme and b.id_author =
                                     a.id_author
group by rollup (t.NameTheme, a.FirstName + ' ' +
a.LastName)
```

### Результат:

	NameTheme	Full Name	Total Cost
1	Computer Science	Donald Knuth	142
2	Computer Science	Herbert Schildt	126
3	Computer Science	Jeffrey Richter	135
4	Computer Science	Matthew MacDonald	135
5	Computer Science	Richard Waymire	253,93
6	Computer Science	NULL	791,93
7	Science Fiction	Clifford Simak	99
8	Science Fiction	Herbert Wells	38
9	Science Fiction	NULL	137
10	Web Technologies	Dino Esposito	81
11	Web Technologies	Matthew MacDonald	46
12	Web Technologies	NULL	127
13	NULL	NULL	1055,93

Итоговая сумма по каждой тематике

Итог по всем тематикам

Как видно из результирующего набора, данный оператор создает дополнительные строки для результирующего запроса, в которые заносит суммарную информацию по нескольким записям с заданной тематикой (t.NameTheme) и именем автора (a.FirstName + " " + a.LastName). Сначала создается новая строка для первого значения тематики (а именно 'Computer Science'). Затем для следующего и т.д. Дополнительная запись (строка) отмечается значением NULL в поле имени автора, а в поле цены отображается сумма значений (результат действия агрегатной функции), для которых тематика равна 'Computer Science'. Эти действия повторяются и для других значений тематики.

В ранних версиях, чтобы получить тот же набор результатов следует воспользоваться набором запросов и объединить их с помощью оператора UNION ALL. Будет выглядеть такой запрос следующим образом:

```
select t.NameTheme, a.FirstName + ' ' + a.LastName as
'FullName', sum(b.Price)
from book.Books b, book.Themes t, book.Authors a
where b.id_theme = t.id_theme and b.id_author =
a.id_author
group by t.NameTheme, a.FirstName + ' ' + a.LastName
UNION ALL
select t.NameTheme, NULL, sum(b.Price)
from book.Books b, book.Themes t, book.Authors a
where b.id_theme = t.id_theme and b.id_author =
a.id_author
group by t.NameTheme
UNION ALL
select NULL, NULL, sum(b.Price)
from book.Books b, book.Themes t, book.Authors a
where b.id_theme = t.id_theme and b.id_author =
a.id_author
```

Результат:

	NameTheme	Full Name	Total Cost
1	Science Fiction	Clifford Simak	99
2	Web Technologies	Dino Esposito	81
3	Computer Science	Donald Knuth	142
4	Computer Science	Herbert Schildt	126
5	Science Fiction	Herbert Wells	38
6	Computer Science	Jeffrey Richter	135
7	Computer Science	Matthew MacDonald	135
8	Web Technologies	Matthew MacDonald	46
9	Computer Science	Richard Waymire	253,93
10	Computer Science	NULL	791,93
11	Science Fiction	NULL	137
12	Web Technologies	NULL	127
13	NULL	NULL	1055,93

Итого по тематикам

Общий итог

Оператор **CUBE** создает надагрегатные строки для всех возможных комбинаций полей GROUP BY. Как и ROLLUP, он рассчитывает текущие суммы или средние значения, но он создает надагрегаты для всех комбинаций, которые не возвращаются оператором ROLLUP. В этом и заключается его отличие.

Чтобы понять лучше вышесказанную разницу, перепишем предыдущий пример с использованием оператора CUBE.

```
select t.NameTheme, a.FirstName + ' ' + a.LastName as
                                'FullName', sum(b.Price)
from book.Books b, book.Themes t, book.Authors a
where b.id_theme = t.id_theme and b.id_author =
                                a.id_author
group by t.NameTheme, a.FirstName + ' ' + a.LastName
with cube;
-- ИЛИ
select t.NameTheme, a.FirstName + ' ' + a.LastName as
                                'FullName', sum(b.Price)
from book.Books b, book.Themes t, book.Authors a
where b.id_theme = t.id_theme and b.id_author =
                                a.id_author
group by cube (t.NameTheme, a.FirstName + ' ' +
                                a.LastName)
```

Результат:

Results		Messages	
	NameTheme	Full Name	Total Cost
5	Computer Science	Donald Knuth	142
6	NULL	Donald Knuth	142
7	Computer Science	Herbert Schildt	126
8	NULL	Herbert Schildt	126
9	Science Fiction	Herbert Wells	38
10	NULL	Herbert Wells	38
11	Computer Science	Jeffrey Richter	135
12	NULL	Jeffrey Richter	135
13	Computer Science	Matthew MacDonald	135
14	Web Technologies	Matthew MacDonald	46
15	NULL	Matthew MacDonald	181
16	Computer Science	Richard Waymire	253,93
17	NULL	Richard Waymire	253,93
18	NULL	NULL	1055,93
19	Computer Science	NULL	791,93
20	Science Fiction	NULL	137
21	Web Technologies	NULL	127

Итого по авторам

Итого по тематикам

В результате работы оператора CUBE создаются дополнительные строки для результирующего запроса, в которые заносится суммарная информация по нескольким записям с заданной тематикой (t.NameTheme) и именем автора (a.FirstName + " + a.LastName). Сначала определяются суммарные стоимости для всех записей с одинаковым значением тематики. В данном примере записи со значениями NULL в поле с именем автора содержат сумму цен по тематикам. Записи со значением NULL в поле с тематикой (t.NameTheme) содержат сумму цен для одинаковых авторов.

Стоит отметить, что при работе оператора CUBE при N атрибутах, результат состоит из 2-х в степени N различных результатов, поэтому оператор CUBE является **очень ресурсоемким** и применяется только при малом количестве атрибутов или малом количестве данных.

Оператор **GROUPING SETS** используют для объединения группирования, поскольку он позволяет одновременно группировать как по уникальным значениям одного атрибута, так и по их комбинациям. Кроме того, он может выступать в качестве замены операторов CUBE и ROLLUP. Для этого необходимо указать все допустимые комбинации агрегаций того или иного оператора в выражении GROUPING SET.

Работу данного оператора рассмотрим для наглядности на том же примере, то есть выведем на экран отчет об общей сумме книг каждой тематики и каждого автора:

```
select t.NameTheme, a.FirstName + ' ' + a.LastName as
'FullName',
       sum(b.Price) as 'Total price'
from book.Books b, book.Themes t, book.Authors a
where b.id_theme = t.id_theme and b.id_author =
       a.id_author
group by grouping sets (t.NameTheme, a.FirstName + ' ' +
a.LastName);
```

Результат:

	NameTheme	Full Name	Total Cost
1	NULL	Clifford Simak	99
2	NULL	Dino Esposito	81
3	NULL	Donald Knuth	142
4	NULL	Herbert Schildt	126
5	NULL	Herbert Wells	38
6	NULL	Jeffrey Richter	135
7	NULL	Matthew MacDonald	181
8	NULL	Richard Waymire	253,93
9	Computer Science	NULL	791,93
10	Science Fiction	NULL	137
11	Web Technologies	NULL	127

Как видно из результата, оператор GROUPING SETS группирует по каждому отдельному полю в списке. Таким образом, вы можете видеть общую стоимость книг для каждой тематики и автора, равносильно работе обычного оператора GROUP BY с той лишь разницей, что в поле, которое не учитывается, указывается значение NULL.

Итак, если наличие всех группам не требуется (как у операторов ROLLUP или CUBE), тогда следует воспользоваться оператором GROUPING SETS, чтобы задать только уникальные группировки.

Но это еще не все. В списке оператора GROUPING SETS можно указывать несколько наборов группирования, разделенных запятыми. В таком случае, все они считаются единым набором, и результат их действий объединяется. Фактически результирующий набор может быть перекрестным объединением (декартовым множеством значений) группирующих наборов.

Например, в приложении **GROUP BY GROUPING SETS ((Column1, Column2), Column3, Column4)** поля Column1 и Column2 будут обработаны как одно поле.

Рассмотрим все на примере:

```
select t.NameTheme, a.FirstName + ' ' + a.LastName as
'FullName',
    sum(b.Price) as 'Total price'
from book.Books b, book.Themes t, book.Authors a
where b.id_theme = t.id_theme and b.id_author =
    a.id_author

group by
    grouping sets ((t.NameTheme, a.FirstName + ' ' +
        a.LastName),
        t.NameTheme);
```

Результатом будет одновременно группирование (тематика, автор) и по тематикам в целом:

	NameTheme	Full Name	Total Cost
1	Computer Science	Donald Knuth	142
2	Computer Science	Herbert Schildt	126
3	Computer Science	Jeffrey Richter	135
4	Computer Science	Matthew MacDonald	135
5	Computer Science	Richard Waymire	253,93
6	Computer Science	NULL	791,93
7	Science Fiction	Clifford Simak	99
8	Science Fiction	Herbert Wells	38
9	Science Fiction	NULL	137
10	Web Technologies	Dino Esposito	81
11	Web Technologies	Matthew MacDonald	46
12	Web Technologies	NULL	127

По тематике  
'Computer Science'  
и каждому автору

По тематике  
'Science Fiction'  
и каждому автору

По тематике  
'Web Technologies'  
и каждому автору

Как уже было выше сказано, оператор GROUPING SETS может давать результат, аналогичный работе операторов ROLLUP или CUBE. Рассмотрим, как это можно сделать, чтобы уметь выбирать лучший и самый простой вариант. Сначала проанализируем сходство операторов CUBE и GROUPING SETS.

Например, напишем запрос, который выведет среднее количество проданных книг за весь период работы издательства в разрезе лет и магазинов, которые реализовывали книги.

```
select convert(char(4), DATEPART(YEAR, s.DateOfSale))
as 'Year',
    sh.NameShop as 'Shop',
    avg(s.Quantity) as 'Average sales'
from book.Books b, sale.Sales s, sale.Shops sh
where b.ID_BOOK=s.ID_BOOK and s.ID_SHOP=sh.ID_SHOP
group by
    cube ((DATEPART(YEAR, s.DateOfSale)),
        sh.NameShop);

-- ИЛИ
select convert(char(4), DATEPART(YEAR, s.DateOfSale)) as
'Year ',
    sh.NameShop as 'Shop',
    avg(s.Quantity) as 'Average sales'
```

```

from book.Books b, sale.Sales s, sale.Shops sh
where b.ID_BOOK=s.ID_BOOK and s.ID_SHOP=sh.ID_SHOP
group by
    grouping sets ( (DATEPART(YEAR, s.DateOfSale),
                    sh.NameShop),
                  (DATEPART(YEAR, s.DateOfSale)),
                  (sh.NameShop),
                    )
    );

```

Результат в обоих случаях будет следующий:

	Year	Shop	Average Sales
1	2010	HashTag	4
2	2016	HashTag	5
3	NULL	HashTag	4
4	2011	Rare Books	5
5	NULL	Rare Books	5
6	2010	Smith&Brown	4
7	NULL	Smith&Brown	4
8	NULL	NULL	4
9	2010	NULL	4
10	2011	NULL	5
11	2016	NULL	5

А теперь сравним, как будут выглядеть эквивалентные запросы с использованием операторов ROLLUP и GROUPING SETS.

```

select convert(char(4), DATEPART(YEAR, s.DateOfSale )) as
'Year',
    sh.NameShop as 'Shop',
    avg(s.Quantity) as 'Average sales'
from book.Books b, sale.Sales s, sale.Shops sh
where b.ID_BOOK=s.ID_BOOK and s.ID_SHOP=sh.ID_SHOP
group by
    rollup ((DATEPART(YEAR, s.DateOfSale)),
            sh.NameShop);

```



```
-- ИЛИ
select convert(char(4), DATEPART(YEAR, s.DateOfSale)) as
'Year',
    sh.NameShop as 'Shop',
    avg(s.Quantity) as 'Average sales'
from book.Books b, sale.Sales s, sale.Shops sh
where b.ID_BOOK=s.ID_BOOK and s.ID_SHOP=sh.ID_SHOP
group by
    grouping sets ( (DATEPART(YEAR, s.DateOfSale),
                    sh.NameShop),
                    (DATEPART(YEAR, s.DateOfSale)),
                    ()
    );
```

Результат работы обоих запросов:

	Year	Shop	Average Sales
1	2010	HashTag	4
2	2010	Smith&Brown	4
3	2010	NULL	4
4	2011	Rare Books	5
5	2011	NULL	5
6	2016	HashTag	5
7	2016	NULL	5
8	NULL	NULL	4

**ПРИМЕЧАНИЕ!** При использовании операторов ROLLUP, CUBE или GROUPING SETS нельзя применять GROUP BY ALL, а в списке GROUP BY не должно быть более 10 полей. Также нельзя использовать данные типа text или image, поскольку они не подлежат упорядочиванию.

## 3. Операторы PIVOT и UNPIVOT

Разного рода магазины и предприятия в ходе своей деятельности используют большую базу данных и время от времени им необходимо получать по этим данным статистику. Например, получить для сравнения отчет по продажам за разные годы. Для решения таких бизнес-задач можно формировать данные в виде **сводных** или **перекрестных таблиц (cross-tabulation)**. Это специальный тип статистического запроса, в котором сгруппированные записи для одного из полей превращаются в отдельные поля.

Для создания сводных таблиц используется оператор **PIVOT**.

```
SELECT поле_для_заголовка_строки,
       [первое_поле_для_значений], ...
FROM ( { название_таблицы | SELECT_запрос }) -- откуда
      получать данные
PIVOT
(
    функция_агрегирования(поле)
    FOR [поле_для_заголовков_столбца]
    IN ( [первое_поле_для_значений], ... )
) AS псевдоним_сводной_таблицы
ORDER BY имя_поля | номер_поля [{ASC | DESC}] --
      обязательная инструкция
```

Чтобы понять лучше работу данного оператора и сам принцип построения сводных таблиц, рассмотрим все по порядку. Для начала напишем запрос, который выводит среднюю стоимость продажи по каждой тематике:

```
select theme.NameTheme as 'Topic',
       AVG(sale.Price*sale.Quantity) as 'Average sales'
from sale.Sales sale, book.Books book, book.Themes theme
where book.ID_THEME=theme.ID_THEME AND book.ID_BOOK=sale.
                                         ID_BOOK
group by theme.NameTheme;
```

Результат:

Results		Messages
	Topic	Average Sales
1	Computer Science	257,5
2	Science Fiction	131,666666666667
3	Web Technologies	350

Данный запрос возвращает две колонки данных: в одном – названия тематик, а в другом – средняя стоимость продажи по каждой теме. Но, иногда пользователю необходимы для наглядности данные в виде сводной таблицы, в которой, например, в одной строке указываются все средние объемы продаж определенных тематик.

Чтобы создать сводную таблицу следует предпринять **следующие действия:**

1. Выбрать необходимые данные с помощью подзапроса, который называют **производной таблицей (derived table)**.
2. Применить оператор PIVOT и указать функцию агрегирования, которую будете использовать.
3. Определить, какие поля будут включены в результирующий набор.

В отличие от обычных сводных таблиц, например, в MS Excel, MS Access (перекрестные таблицы) и т.д., оператор PIVOT в SQL Server требует явно перечислять поля для результирующего набора. Это является жестким ограничением, поскольку для этого необходимо знать характер данных, с которыми вы работаете.

Итак, перепишем вышерассмотренный пример таким образом, чтобы образовалась сводная таблица, которая будет отображать данные для сравнения о среднем объеме продаж учебников и книг тематик 'Computer Science', 'Science Fiction' и 'Web Technologies'.

```
select 'Average Sales',
      [Computer Science], [Science Fiction], [Web
Technologies]
from (select t.NameTheme, (s.price*s.quantity) as 'Cost'
      from Sales s, Books b, Themes t
      where b.id_theme = t.id and
            s.id_book = b.id) as SourceTable
pivot (
  avg(Cost) for NameTheme in ([Computer Science], [Science
Fiction],
  [Web Technologies])) as PivotTable;
```

Результат:

Results		Messages		
	(No column name)	Computer Science	Science Fiction	Web Technologies
1	Average Sales	257,5	131,666666666667	350

У нас образовалась одноуровневая сводная таблица. Построение двухуровневой таблицы несколько сложнее. Для примера рассмотрим построение сводной таблицы, в которой будут отображаться данные о количестве

продаж магазинами четырех определенных книг. Образованные данные следует отсортировать по магазинам.

```
select pvt.id_shop,
       [1] as 'Ring Around the Sun',
       [5] as 'CLR via C#',
       [8] as 'The Art of Computer Programming, vol.1 ',
       [14] as 'Swing: A Beginner Guide'
from (select id_book, id_shop, id from Sales) p
pivot
(count(id) for id_book in ([1],[5],[8],[14])) as pvt
order by pvt.id_shop;
```

Оператор UNPIVOT выполняет действия, обратные по отношению к операции PIVOT, то есть преобразует данные, которые имеют вид сводных таблиц, то есть записанные в одну строку, в столбец. Этот оператор очень полезен для нормализации таблиц, в которых существует несколько полей с одинаковым типом данных.

Результат:

Results Messages			
id_book	id_shop	id	
1	5	1	
2	5	2	
3	10	3	
4	10	4	
5	10	5	
6	10	6	
7	10	7	
8	7	8	
9	7	9	
10	7	10	
11	7	11	
12	6	12	
13	6	13	
14	6	14	

Results Messages				
id_shop	Ring Around the Sun	CLR via C#	The Art of Computer Programming, vol.1	Swing: A Beginner Guide
1	1	0	0	0
2	0	0	1	0
3	0	1	0	1
4	0	0	0	0

ib\_book = 1

ib\_book = 5

ib\_book = 8

ib\_book = 14

книга

магазин

Для демонстрации работы данного оператора, применим оператор UNPIVOT для преобразования сводной таблицы с данными о среднем объеме продаж 3-х тематик.

```
select NameTheme, Cost
from (      'Average Sales',
    [Computer Science], [Science Fiction], [Web
Technologies]
    from (select t.NameTheme, (s.price*s.quantity) as
'Cost'
        from Sales s, Books b, Themes t
        where b.id_theme = t.id and
        s.id_book = b.id) as SourceTable
pivot (
avg(Cost) for NameTheme in ([Computer Science], [Science
Fiction],
    [Web Technologies])) as PivotTable) as Pvt
unpivot
(Cost for NameTheme in ([Computer Science], [Science
Fiction],
    [Web Technologies])) as UnpivotTable;
```

Результат:

	NameTheme	(No column name)
1	Computer Science	891,93
2	Science Fiction	137
3	Web Technologies	127

## 4. Представления в MS SQL Server

**Представление (view)** – это объект БД, который имеет внешний вид таблицы, но в отличие от нее не имеет своих собственных данных. Представление лишь предоставляет доступ к данным одной или нескольких таблиц, на которых оно базируется. С помощью представления можно осуществлять контроль данных, вводимых пользователем, а также упростить работу разработчиков: запросы, которые чаще всего выполняются, можно поместить в представление, чтобы не писать один и тот же код постоянно.

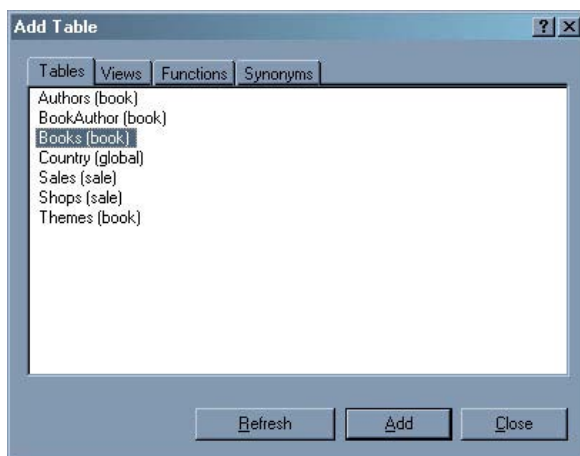
Поскольку представления являются объектом базы данных, то информация о них хранится в системной таблице **sysobjects**, текст оператора записывается в таблицу **syscomments**, а физическое размещение находится в папке View. Кроме того, автоматически заполняются следующие представления: **sys.views**, **sys.columns**, **sys.sql\_expression\_dependencies** и **sys.sql\_modules** (содержится текст инструкции create view).

Представление можно создавать как с помощью Management Studio, так и с помощью SQL. Рассмотрим оба способа.

Итак, создадим представление, что позволяет получить информацию о книгах, тираж которых более 3000, и их авторах.

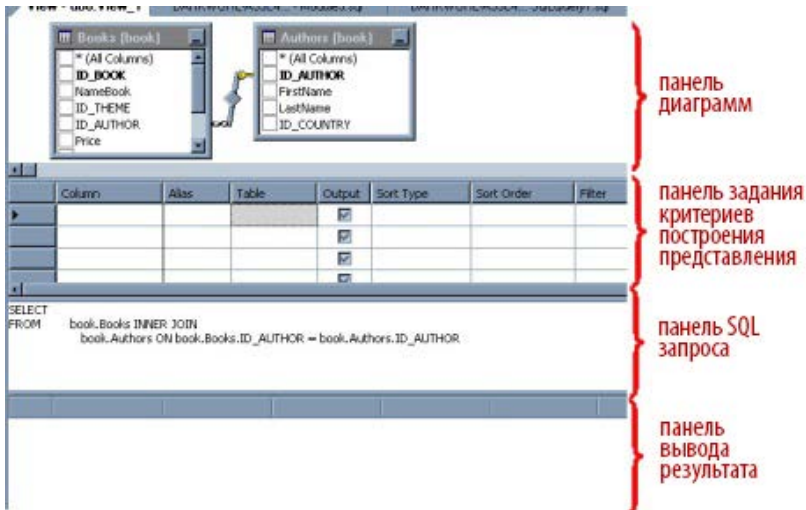


Для того, чтобы создать такой запрос средствами Management Studio, а именно с помощью Create View Wizard, нужно в контекстном меню папки View выбрать пункт **New view**. После этого перед Вами появится окно с предложением выбрать те таблицы, из которых будет браться информация для представления:



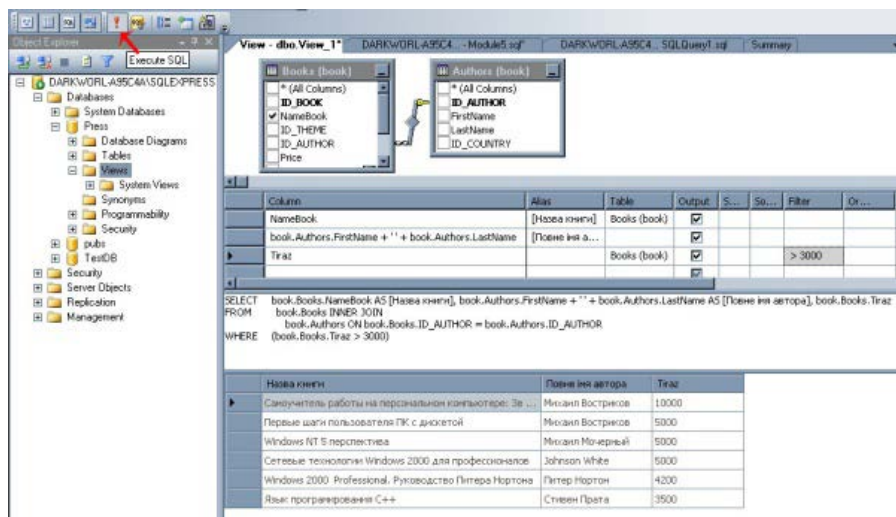


Выбрав таблицы, которые Вам нужны (в нашем случае это Books и Authors) нажимаем Add и Close. После этого появится окно конструктора представлений **View Designer**:



Итак, в поле Table выбираем нужные таблицы, в поле Column – необходимые поля этой таблицы для вывода. В поле Alias можно задать псевдонимы на поля и тому подобное.

После проделанной работы и выполнения запроса мы получим следующий результат:



После этого представления можно сохранить.

Для создания представлений средствами SQL используют инструкцию **CREATE VIEW**, которая имеет следующий синтаксис:

```
CREATE VIEW [схема.] название_представления [ (поле [, ...
n] ) ]
[ WITH ENCRYPTION | SCHEMABINDING | VIEW_METADATA ]
AS
<оператор SELECT>
[ WITH CHECK OPTION];
```

Расшифруем коротко каждый из параметров данного оператора:

- **WITH ENCRYPTION** – указывает на то, что представление будет сохраняться в системной таблице syscomments в зашифрованном виде. Расшифровать такое представление невозможно, поэтому при выборе данного параметра следует убедиться, что у вас есть его копия;

- **WITH SCHEMABINDING**. Эта опция устанавливает запрет удаления таблиц, представлений или функций, на которые ссылается представление, раньше, чем оно будет удалено;
- **WITH VIEW\_METADATA** – указывает на то, что представление в режиме просмотра будет возвращать его метаданные, то есть информацию о его структуре, а не записи таблиц;
- **WITH CHECK OPTION** – создает модифицированное представление, в котором существует возможность запретить выполнение операций INSERT или DELETE, если при этом нарушается условие, заданное в конструкции WHERE.

После ключевого слова AS должен размещаться запрос с использованием оператора SELECT, предназначенный для выбора данных, которые будут включены в результирующее представление. Стоит отметить, что для того, чтобы изменить произвольное представление, необходимо его пересоздать, то есть удалить и снова создать. А при удалении нужно удалить все зависящие от него объекты – триггеры, хранимой процедуры и т.п. Это является одним из основных неудобств при работе с представлениями.

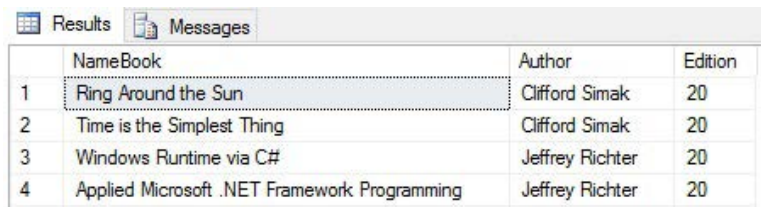
Итак, попробуем написать представление, которое мы создавали с помощью View Designer, используя оператор SQL CREATE VIEW:

```
create view BooksEdition_View2 (NameBook, Author, Edition)
as
select b.NameBook, a.FirstName + ' ' + a.LastName as
'FullName', b. Quantity
from Books b, Authors a
where b.id_author = a.id and b.quantity > 10;
```

Созданное представление можно использовать как любую таблицу базы данных. И чтобы получить из него данные, следует написать простой запрос на выборку с помощью оператора SELECT:

```
select *
from book.BooksEdition_View2;
```

Результат:



	NameBook	Author	Edition
1	Ring Around the Sun	Clifford Simak	20
2	Time is the Simplest Thing	Clifford Simak	20
3	Windows Runtime via C#	Jeffrey Richter	20
4	Applied Microsoft .NET Framework Programming	Jeffrey Richter	20

Но само по себе представление НЕ СОДЕРЖИТ никаких данных. Оно просто ссылается на уже существующие данные, то есть является обычным SELECT запросом, который просто имеет имя. Фактически, когда с помощью запроса вы обращаетесь к представлению, оптимизатор запросов заменяет эту ссылку на описание представления (тело), а затем создает план выполнения.

Представления могут быть основаны на данных из нескольких таблиц и даже на основе других представлений. Также представление могут содержать данные, полученные на основе различных выражений, в том числе и на основе функций агрегирования.

Различают следующие **типы представлений**:

1. **Обычные представления на основе объединений** – это представление на выборку данных.

2. **Модифицированные (обновляемые) представления** – это представления, которые поддерживают модификацию данных.
3. **Индексированные или материализованные представления** – это представления с использованием кластеризованного индекса. Такие представления позволяют существенно повысить производительность даже очень сложных запросов. Индексированные представления подробнее будут рассмотрены при изучении индексов.
4. **Секционированные представления** – это представления, которые при создании используют оператор UNION ALL для объединения нескольких запросов, которые построены на основе таблиц с одинаковой структурой и хранятся либо в одном экземпляре SQL Server, или в группе автономных экземпляров SQL Server, которые называются федеративными серверами баз данных. Подробнее информацию можно прочитать в разделе "Создание секционированных представлений" MSDN.

Для успешной работы стоит только понять, что в основе представления лежат обычные запросы на выборку. Итак, все, что можно делать с запросами относится и к представлениям. И, если необходимо вывести данные в отсортированном порядке нам не удастся, поскольку сортировать можно только результирующие данные представления:

```
select *  
from book.BooksEdition_View2  
order by 1;
```

Хотя и здесь существует нюанс: если в операторе SELECT используется опция TOP, тогда инструкция ORDER BY может использоваться при создании представления. То есть следующий запрос будет верным:

```
create view TestView
as
select top (3) b.NameBook, t.NameTheme, b.DateOfPublish
from book.Books b, book.Themes t
where b.ID_THEME = t.ID_THEME order by 1;
```

Несмотря на удобство использования представлений, кроме ограничения на использование выражения ORDER BY существует еще ряд существенных ограничений:

- нельзя использовать в качестве источника данных набор, полученный в результате выполнения хранимых процедур;
- при создании представления оператор SELECT не должен содержать операторы COMPUTE или COMPUTE BY;
- представление не может ссылаться на временные таблицы, поэтому в операторе создания ЗАПРЕЩЕНО использовать оператор SELECT INTO;
- данные, которые используются представлением, не сохраняются отдельно, поэтому при изменении данных представления меняются данные базовых таблиц;
- представление не может ссылаться больше, чем на 1024 поля;
- UNION и UNION ALL недопустимы при формировании представлений.

Для модификации представления используется оператор **ALTER VIEW**:

```
ALTER VIEW [схема.] название_представления [ (поле [, ...
n] ) ]
[ WITH ENCRYPTION | SCHEMABINDING | VIEW_METADATA ]
AS
<оператор SELECT>
[ WITH CHECK OPTION];
```

Для того, чтобы удалить представления используется оператор **DROP VIEW**:

```
DROP VIEW название_представления;
```

В начале данного раздела мы говорили, что представления могут быть **модифицируемыми (обновляемыми, updateable view)**. Что же это за представления? Как видно из самого названия, такие представления позволяют не только читать данные, но и изменять их. Модифицируемыми (обновляемыми) представления становятся, если при их создании указать инструкцию **WITH CHECK OPTIONS**.

К таким представлениям можно применять команды INSERT, UPDATE и DELETE. Причем они позволяют запретить выполнение операций INSERT или DELETE, если при этом нарушается условие, заданное в конструкции WHERE. Это можно объяснить так: если новая запись, который вставляется пользователем или полученный в результате обновления существующей записи, не удовлетворяет условию запроса, то вставка этой записи будет отменена и возникнет ошибка.

Рассмотрим **условия создания модифицированных представлений**:

- все модификации должны касаться только одной таблицы, то есть модифицированные представления строятся только на однотабличных запросах;
- все изменения должны касаться только полей таблицы, а не производных полей. То есть, нельзя модифицировать поля, полученные:
  - с помощью агрегатной функции: AVG, COUNT, SUM, MIN, MAX, GROUPING, STDEV, STDEVP, VAR и VARP;
  - на основе расчетов с участием других полей или операций над полем, например substring. Поля, сформированные с помощью операторов UNION, UNION ALL, CROSSJOIN, EXCEPT и INTERSECT также считаются расчетными и также не могут обновляться.
- при определении представления нельзя использовать инструкции GROUP BY, HAVING и DISTINCT;
- нельзя использовать опцию TOP вместе с инструкцией WITH CHECK OPTION, даже в подзапросах.

Если указанные ограничения не позволяют модифицировать данные, тогда можно воспользоваться триггерами INSTEAD OF или секционированными представлениями.

Например, необходимо иметь представление, которое будет отображать информацию о книгах, которые имели тираж более 10 экземпляров:

```
create view Books10 (NameBook, Price, Quantity) as
select NameBook, Price, Quantity
from Books
where quantity > 10
with check option;
```

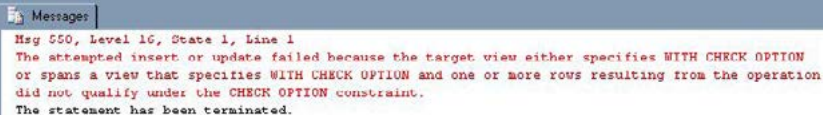


Само по себе модифицированное представление ничем не отличается от обычного, но попробуем проверить данное модифицированное представление на операцию вставки данных.

```
insert into Books5000 (NameTovar, Price, DrawingOfBook)
values ('Microsoft SQL Server 2000 за 21 день', 350.50,
3000);
```

В случае обычного представления, пользователь может добавить такую книгу и ошибка при этом не возникнет. Но данный запрос на вставку является некорректным и приведет к путанице: представление при следующем обращении введенных данных показывать не будет, поскольку они не соответствуют критерию отбора данных.

При введении таких данных в модифицированное представление, сервер сразу выдаст сообщение об ошибке, поскольку данные ввода не соответствуют основному условию отбора (тираж более 10 экземпляров).



```
Msg 550, Level 16, State 1, Line 1
The attempted insert or update failed because the target view either specifies WITH CHECK OPTION
or spans a view that specifies WITH CHECK OPTION and one or more rows resulting from the operation
did not qualify under the CHECK OPTION constraint.
The statement has been terminated.
```

Следующий запрос на вставку является корректным и выполняется, поскольку тираж выпуска указывается более 10.

```
Insert into Books10 (NameBook, Price, Quantity)
values
('Modern Web Development', 35, 50);
```

Подытоживая, следует сказать, что хотя модифицированные представления и используются для изменения данных, но для этой цели они практически никогда не используются. Лучше всего для таких действий подходят хранимые процедуры. Кроме того, они гораздо более гибкие.

## 5. Домашнее задание

1. С помощью оператора `COMPUTE BY` написать запрос, который выводит названия магазинов и суммарное количество заказанных магазином книг.
2. С помощью оператора `COMPUTE` вывести данные о том, сколько получило издательство от продажи книг за последний год.
3. С помощью операторов `CUBE` и `ROLLUP` написать два запроса, которые будут отображать информацию о среднем количестве продажи книг магазинами Великобритании и Канады в промежутке 01/01/2008 по 01/09/2008. Показать среднее количество продаж, как по каждой стране, так и по отдельному магазину.
4. Используя оператор `GROUPING SET` написать запрос, который выведет максимальный тираж книг за весь период работы издательства в разрезе авторов и годов выпуска книг.
5. Используя оператор `PIVOT` создать сводную таблицу, отражающую минимальные цены продажи книг отдельными магазинами за последний год.
6. Используя оператор `PIVOT`, создать двухуровневую сводную таблицу, которая отражает данные о количестве выпуска издательством книг всех тематик на протяжении трех выбранных лет. Отсортировать данные по тематикам.

7. С помощью оператора UNPIVOT превратить сводные таблицы в обычные для примеров (5) и (6).
8. Найти авторов, живущих в тех странах, где есть хотя бы один из магазинов по распространению книг, занесенных в БД. Результат запроса поместить в отдельное представление.
9. Написать представления, содержащие самую дорогую книгу тематики, например, "Web Technologies".
10. Написать представление, которое позволяет вывести всю информацию о работе магазинов. Отсортировать выборку по странам в возрастающем и по названиям магазинов в убывающем порядке.
11. Написать зашифрованное представление, показывающее самую популярную книгу.
12. Написать модифицированное представление, в котором предоставляется информация об авторах, имена которых начинаются с А или В.
13. Написать представление, в котором с помощью подзапросов вывести названия магазинов, которые еще не продают книги вашего издательства.





## Урок №5

# 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.