

Урок №5

Использование PIVOT, UNPIVOT. Обработка ошибок

Содержание

| 1. | Директивы COMPUTE и COMPUTE BY3 |
|----|--|
| 2. | Надагрегатные операторы ROLLUP, CUBE и GROUPING SETS |
| 3. | Операторы PIVOT и UNPIVOT18 |

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]

[ ВУ выражение [ ,...n ] ]
]
```

Оператор **СОМРИТЕ** 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.

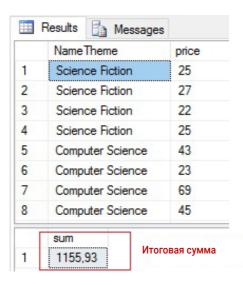
```
select t.NameTheme, sum(b.Price)
from book.Books b, book.Themes t
where b.id_theme = t.id_theme
group by t.NameTheme;
```

Результат:

| | NT | (NI= == I === == ==) |
|---|------------------|-----------------------|
| | 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 сортирповку по произвольному полю
compute sum(b.Price);
```



При использовании оператора СОМРИТЕ ВУ в запрос ОБЯЗАТЕЛЬНО включается директива ORDER ВУ. При этом, если ORDER ВУ имеет вид:

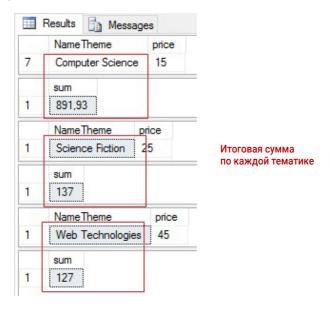
```
ORDER BY t.NameTheme, b.NameBook
    то COMPUTE BY должен быть одним из следующих вариантов
-- 1 вариант
    COMPUTE функция_агрегирования(поле)
    BY t.NameTheme, b.NameBook
-- 2 вариант
    COMPUTE функция_агрегирования(поле)
    BY t.NameTheme
```

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 СОМРИТЕ и СОМРИТЕ ВУ создают новые записи (строки) нереляционных данных. При использовании вышеописанных операторов также нельзя использовать данные типа text или image, поскольку они не подлежат упорядочиванию.



2. Надагрегатные операторы ROLLUP, CUBE и GROUPING SETS

Операторы ROLLUP, CUBE GROUPING SETS задекларированны стандартом ANSI / ISO SQL'99 и используются для создания дополнительных строк в результате выполнения команды SELECT. Данные операторы называют "надагрегатными", 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 з WITH
```

Стоит отметить, что поддержку ANSI стандарта для операторов ROLLUP, CUBE и GROUPING SETS Microsoft реализовал только в верисии 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 надагрегатов. Напишем запрос, в результате которого получим общую сумму книг каждой тематики и каждого автора.

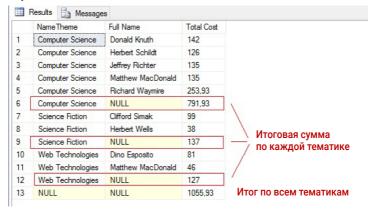
Результат:

| | Name Theme | 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:

2. Надагрегатные операторы ROLLUP, CUBE и GROUPING SETS

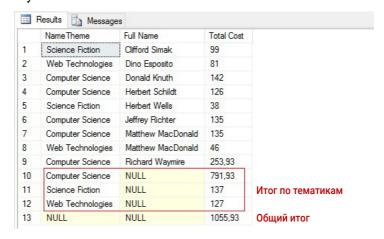
Результат:



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
```



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.

| | 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 | Утог по авторам |
| 11 | Computer Science | Jeffrey Richter | 135 | A VIII III abropaii |
| 12 | NULL | Jeffrey Richter | 135 | |
| 13 | Computer Science | Matthew MacDonald | 135 | |
| 14 | Web Technologies | Matthew MacDonald | 46 | // |
| 15 | NULL | Matthew MacDonald | 181 | 11 |
| 16 | Computer Science | Richard Waymire | 253,93 | |
| 17 | NULL | Richard Waymire | 253,93 | 1 |
| 18 | NULL | NULL | 1055,93 | |
| 19 | Computer Science | NULL | 791,93 | |
| 20 | Science Fiction | NULL | 137 | Итог по тематикам |
| 21 | Web Technologies | NULL | 127 | |

В результате работы оператора СUBE 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 СUBE при N атрибутах, результат состоит из 2-х в степени N различных результатов, поэтому оператор СUBE 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 значением 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:

| | 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 ((Colum1, Column2), Column3, Column4)** поля Column1 и Column2 будут обработаны как одно поле.

Рассмотрим все на примере:

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 | По тематике |
| 3 | Computer Science | Jeffrey Richter | 135 | 'Computer Science' |
| 4 | Computer Science | Matthew MacDonald | 135 | и каждому автору |
| 5 | Computer Science | Richard Waymire | 253,93 | |
| 6 | Computer Science | NULL | 791,93 | По тематике |
| 7 | Science Fiction | Clifford Simak | 99 | 'Science Fiction' |
| 8 | Science Fiction | Herbert Wells | 38 | и каждому автору |
| 9 | Science Fiction | NULL | 137 | , ., |
| 10 | Web Technologies | Dino Esposito | 81 | По тематике |
| 11 | Web Technologies | Matthew MacDonald | 46 | 'Web Technologies |
| 12 | Web Technologies | NULL | 127 | 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'
```

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 |

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

2. Надагрегатные операторы ROLLUP, CUBE и GROUPING SETS

```
-- 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 упорядочеванию.

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 сгруппированые 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_агрегувания (поле)

      FOR [поле_для_заголовков_столбца]

      IN ( [первое_поле_для_значений], ... )

      ) AS псевдоним_сводной_таблицы

      ОRDER 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 объем 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:

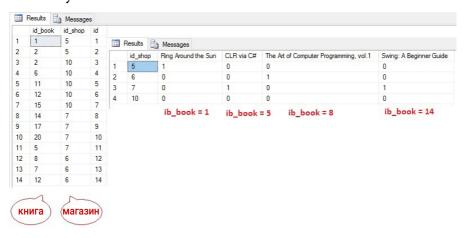


У нас образовалась одноуровневая сводная таблица. Построение двухуровневой таблицы несколько сложнее. Для примера рассмотрим построение сводной таблицы, в которой будут отображаться данные о количестве

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

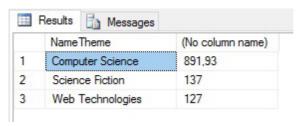
```
select pvt.id_shop,
  [1] as 'Ring Around the Sun',
  [5] as 'CLR via C#',
  [8] as 'The Art of Computer Programming, vol.1 ',
  [14] as 'Swing: A Beginner Guide'
from (select id_book, id_shop, id from Sales) p
pivot
(count(id) for id_book in ([1],[5],[8],[14])) as pvt
order by pvt.id_shop;
```

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.



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

```
select NameTheme, Cost
            '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;
```





Урок №5 **Использование PIVOT, UNPIVOT. Обработка ошибок**

© 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.