**Глава 24. Нюансы функций CALCULATE() и FILTER()**

Это продолжение перевода книги Роб Колли. Формулы DAX для Power Pivot. Главы не являются независимыми, поэтому рекомендую начать сначала.

[Предыдущая глава](http://baguzin.ru/wp/?p=20521) [Содержание](http://baguzin.ru/wp/?p=19780#content) Следующая глава

Как и предыдущая глава, эта не является обязательной. Она раскрывает несколько тонких возможностей функций CALCULATE() и FILTER().

О том, что аргумент <filter> функции CALCULATE() переопределяет контекст фильтра, заданный координатами сводной таблицы, мы уже говорили в [главе 8](http://baguzin.ru/wp/?p=19846). Обсудим это подробнее. Для облегчения восприятия воспользуемся визуальным подходом. Предположим, у нас есть мера – Красные велосипеда, купленные женщинами:

Red Bikes bought by Females :=
CALCULATE (
    SUM(Sales[OrderQuantity]);
    Products[Color] = "Red";
    Products[Category] = "Bikes";
    Customers[Gender] = "F"
)

В сводной таблице установлены фильтры по цвету, полу и году:



Рис. 24.1. Контекст фильтра сводной таблицы

Давайте попробуем понять контекст фильтра в ячейке С16. Координаты сводной таблицы можно рассматривать как набор фильтров для отдельных столбцов одной или нескольких таблиц Power Pivot:



Рис. 24.2. Входящие фильтры таблиц Power Pivot, поступающие из сводной таблицы

Теперь давайте наложим на них фильтры, добавленные (переопределенные) в функции CALCULATE():



Рис. 24.3. Фильтры функции CALCULATE переопределяют фильтры, поступающие из сводной таблицы

Видно, что обе системы фильтров работают с одним и тем же набором Таблица[Столбец]. Фильтры функции CALCULATE переопределяют (замещают) входящие фильтры по столбцам Product[Color] и Customer[Gender], и создают новый фильтр в столбце Product[Category]. Исходные фильтры из сводной таблицы, которые не конфликтуют с фильтрами функции CALCULATE, проходят через меру без изменений. В нашем случае это Calendar[Year].

Давайте еще раз рассмотрим аргумент <filter> функции CALCULATE. Синтаксис функции

CALCULATE(<expression>;<filter1>;<filter2>...),

где <filter1>;<filter2>;... – список ИСТИНА/ЛОЖЬ выражений или таблиц.

В [глава 8. DAX-функция CALCULATE()](http://baguzin.ru/wp/?p=19846) мы начали с использования простых выражений ИСТИНА/ЛОЖЬ, например,

CALCULATE([Total Sales]; Sales[Year] = 2002)

Выражение ИСТИНА/ЛОЖЬ для функции CALCULATE имеет вид

Таблица[Столбец] <оператор> фиксированное значение

где <оператор> – это один из операторов сравнения, таких как =, >, <, <=, >=, <>

Но затем в качестве аргументов <filter> функции CALCULATE мы начали использовать функции ALL(), FILTER(), DATESYTD() и т.д. Эти функции определенно не возвращают ИСТИНА/ЛОЖЬ, так что же они... являются таблицами?

ALL() – это функция, которая говорит «удалить все фильтры», но она также является таблицей. Или другая известная нам функция – COUNTROWS. Она имеет единственный аргумент – <table>. Принимая таблицу в качестве аргумента COUNTROWS подсчитывает число строк в этой таблице. Ровно такой же смысл заложен в аргументе <filter> функции CALCULATE. Она принимает таблицу, и по ее строкам проводит вычисления.

Так, можем ли мы передать ALL() в качестве аргумента COUNTROWS? Еще как можем! А также DATESBETWEEN и VALUES (почему мы назвали именно эти функции? Просто в качестве примера). Определим несколько мер:

Count Days All Calendar := COUNTROWS(ALL(Calendar))

Count Me Some Days := COUNTROWS (
 DATESBETWEEN(Calendar[Date]; "01.02.2002", "01.03.2002" )
)

Count All Months := COUNTROWS(ALL(Calendar[MonthName]))

Count Included Months := COUNTROWS(VALUES(Calendar[MonthName]))



Рис. 24.4. COUNTROWS принимает <table> в качестве аргумента, следовательно, можно использовать функции, возвращающие таблицу

### Существуют десятки функций двойного назначения

ALL, FILTER, TOPN, DATESBETWEEN, DATEADD, DATESYTD, VALUES и многие другие возвращают таблицу, и могут использоваться в любом месте, где формула DAX обращается к таблице. А также эти функции могут использоваться в качестве аргумента <filter> функции CALCULATE. Вот почему мы называем их функциями двойного назначения.

Можно переписать первую формулу выше (и все остальные), используя CALCULATE:

Count Days All Calendar := COUNTROWS(ALL(Calendar)) –>

Count Days All Calendar using CALCULATE := CALCULATE(COUNTROWS(Calendar), ALL(Calendar))



Рис. 24.5. Два способа получить один и тот же результат: использовать ALL() в качестве аргумента <table> или аргумента <filter>

Microsoft могла бы создать отдельные функции для каждой цели, например, ALLFILTER и ALLTABLE, и позволить использовать их только в одном месте (первую как <фильтр> для CALCULATE, а вторую как <таблица>), но вместо того, чтобы удлинять список функций почти вдвое, они позволили нам использовать функции в двух местах.

### Вложенные табличные функции

В большинстве примеров мы видели, как FILTER сокращает набор строк, над которыми мы выполняем операции. Но FILTER можно использовать и для увеличения числа строк по сравнению с тем, что изначально пришло из координат сводной таблицы.

Посмотри на эту бессмысленную меру, созданную для примера:

[Transactions Silly] = CALCULATE([Transactions]; FILTER(Products;1))



Рис. 24.6. Поскольку 1 всегда возвращает ИСТИНА, фильтр ничего не фильтрует; все строки проходят тест

Функция FILTER принимает <table> в качестве своего первого аргумента:

FILTER (<таблица>; <фильтр>)

Таким образом, мы можем передать таблицу или табличную функцию в качестве первого аргумента для функции FILTER, которая сама является табличной функцией! Это увеличит ваши возможности написания формул, как только вы переварите этот факт.

Поскольку ALL() это табличная функция, давайте попробуем:-

[Transactions All Products] =

CALCULATE([Transactions]; FILTER(ALL( Products);1))

Действительно эта мера снимает фильтры с таблицы Product:



Рис. 24.7. FILTER может выполнять функцию «снять фильтры»

Разберем работу меры по шагам:

1. ALL(Products) создает виртуальную копию таблицы Products без каких бы то ни было фильтров.
2. FILTER(ALL(Products);1) проходит через эту виртуальную таблицу по одной строке за раз и оценивает второй аргумент. Поскольку единица всегда истинна, все строк виртуальной таблицы сохраняется. Функция FILTER возвращает другую виртуальную таблицу, содержащую все строкой Products.
3. Наконец, поскольку FILTER (на шаге 2) использовался в качестве аргумента <filter> для CALCULATE, и такие аргументы <filter> переопределяют контекст фильтра, поступающий из сводной, мы получаем количество транзакций для всех продуктов.

Т.е., мы использовали функцию FILTER, чтобы снять все фильтры.

### Собираем все вместе

Давайте вернемся к первому примеру функции CALCULATE() из [главы 8](http://baguzin.ru/wp/?p=19846):

[2002 Sales] = CALCULATE([Total Sales]; Sales[Year] = 2002)

В этой формуле мы использовали в качестве аргумента <filter>, выражение Sales[Year] = 2002, возвращающее ИСТИНА/ЛОЖЬ. Перепишем меру с помощью табличного выражения:

[2002 Sales via FILTER] =

CALCULATE (

[Total Sales];

FILTER(Sales; Sales[Year] = 2002)

)



Рис. 24.8. Наша первая попытка с функцией FILTER дала иные результаты

Меры [2002 Sales] и [2002 Sales via FILTER] дают разные результаты. Почему? Потому что CALCULATE отменяет (игнорирует) входящий контекст фильтра. Таким образом, первая мера просто суммирует продажи за 2002 г., и возвращает это значение во все ячейки сводной таблицы (Н3:Н6). Вторая мера, сначала отрабатывает FILTER, который для всех годов, кроме 2002, возвращает пустую таблицу. Таким образом сумма появляется только в 2002 г., в остальных годах сумма равна нулю (I3:I6).

Чтобы мера давала требуемый результат, ее нужно усовершенствовать:

[2002 Sales via FILTER ALL Sales Year] =

CALCULATE (

[Total Sales];

FILTER(ALL(Sales[Year]); Sales[Year] = 2002)

)



Рис. 24.9. Вторая попытка оказалась удачной

Эти ухищрения сами по себе не приносят полезны. Наша цель – проиллюстрировать концепцию и подготовить почву для серьезной работы. Способность вложить одну табличную функцию в другую оказывается невероятно мощной, как и способность переключаться между их использованием в качестве <table> и в качестве <filter>.