**Глава 16. SUMX() и другие X функции (итераторы)**

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=20026) [Содержание](http://baguzin.ru/wp/?p=19780#content) Следующая глава

[Ранее](http://baguzin.ru/wp/?p=19780#SalperDay) мы ввели меру [Sales per Day]:



Рис. 16.1. Промежуточные итоги не соответствуют сумме входящих в них частей

В то время, как ваши меры будут усложняться, такое несоответствие будет происходить все чаще и чаще. В нашем примере это происходит потому, что мера [Sales per Day] имеет различный знаменатель для каждой модели – число дней продаж. Аналогично, если у вас есть средняя температура для 12 месяцев в году, усреднение этих чисел не даст вам среднюю температуру за год, так как каждый месяц состоит из разного количества дней.

В общем случае правильная логика для наименьшей детализации не всегда является правильной для следующего уровня. Другими словами, иногда вы нуждаетесь в том, чтобы сумма (или среднее) итогов равнялось сумме составляющих. SUMX() и другие подобные X-функции помогут вам в этом.

Функция SUMX() имеет два аргумента (рис. 16.2):

SUMX(<table or table expressions>; <arithmetic expression>)



Рис. 16.2. Синтаксис функции SUMX()

SUMX() работает следующим образом:

1. Она идет по каждой строке таблицы, беря одну за раз. В первый аргумент можно передать имя таблицы или использовать функцию, возвращающую таблицу, например VALUES() или FILTER(). Содержимое таблицы зависит от контекста фильтра текущей ячейки меры. Такое пошаговое поведения часто описывают как перебор или итерации.
2. Для каждой строки вычисляется арифметическое выражение с использованием контекста фильтра текущей строки.
3. Функция запоминает результат арифметического выражения из каждой строки, и когда все строки просмотрены возвращает сумму арифметических выражений.

### SUMX() в действии

Возвращаясь к рис. 16.1 напишем новую меру:

[Sales per Day Totals Add Up] =

SUMX(VALUES(Products[ModelName]);

[Sales per Day])

)

Обратите внимание, что мы использовали VALUES(Products[ModelName]) для аргумента <таблица>. Это позволяет нам быть очень точными – мы хотим, чтобы SUMX() прошла через все уникальные значения ModelName из текущего контекста фильтра. Если же указать всю таблицу Products (и не использовать функцию VALUES), SUMX() будет проходить через каждую строку таблицы Products из текущего контекста фильтра. В первом и втором случае мы получим разное число строк в аргументе <таблица>.



Рис. 16.3. Для новой меры значение итогов равно сумме значений для отдельных моделей

### Остановимся на работе функции SUMX() подробнее

Рассмотрим, как получилась сумма $ 21 600 на рисунке 16.3 в соответствии с тремя пунктами, изложенными выше. Контекст фильтра в данном случае дает полностью нефильтрованную таблицу Products, так как ячейка D3 имеет фильтры: Year=2001, Products=All. Таким образом, VALUES(Products[ModelName]) возвращает каждое уникальное значение столбца [ModelName] из таблицы Products. Таких значений 119. Проверим, введя меру

[ModelName Values] = COUNTROWS(VALUES(Products[ModelName]))



Рис. 16.4. Есть 119 различных моделей, хотя на рис. 16.3 представлены только 3

Почему 119 против 3? Всего моделей 119, но только три из них имели продажи в 2001 г. Т.е., только три возвращают непустые результаты для [Sales per day], что и показывает сводная на рис. 16.3.

На втором шаге для каждого из этих 119 значений SUMX() оценивает меру [Sales per Day]. Контекст фильтра Year=2001 сохраняется на протяжении всего процесса для каждой строки. Но контекст фильтра Products[ModelName] меняется каждый раз, когда SUMX() переходит к следующей из 119 строк. Таким образом, сначала SUMX() оценивает [Sales per Day] с контекстом фильтра Year=2001, ModelName="All-Purpose Bike Stand", и возвращает ноль, потому что продаж этой модели в 2001 году не было. Затем SUMX() переходит к Year=2001, ModelName="Bike Wash", и т.д. по всем 119 моделям. Только три из этих 119 строк в VALUES(Products[ModelName]) возвращают непустые результаты для [Sales per day], и это три, которые мы видели на рис. 16.3: "Mountain-100", "Road-150" и "Road-650".

На третьем шаге SUMX() суммирует все 119 результатов [Sales per day]: 116 пустых значений и три непустых, что в итоге дает $21 600.

### MIN(), MAX() или AVERAGE()

Есть еще три функции, которые действуют точно так же, как SUMX(). Единственное различие заключается в том, что на последнем шаге вместо суммирования всех результатов, возвращаемых каждым шагом, они затем применяют другую агрегацию: MIN(), MAX() или AVERAGE().

### FILTER()

Функцию FILTER следовало бы назвать FILTERX, так как она принадлежит семейству функций X. Их ключевое отличие заключается в том, что фильтр возвращает таблицу, а не одно значение, такое как SUMX (или другие функции X). Однако принципы их работы и синтаксис аналогичны:

SUMX(<table or table expression>; <arithmetic expression>)

FILTER(<table or table expression>; <filter condition>)

Обе функции перебирают таблицу (первый аргумент) строка за строкой. Далее SUMX вычисляет <arithmetic expression>), а FILTER – <filter condition> для каждой строки. Наконец SUMX суммирует все значения, вычисленные в каждой строке, а FILTER собирает все строки таблицы, удовлетворяющие <filter condition>, и возвращает их в виде таблицы.

Мы надеемся, что вы увидите здесь сходство.

### STDEVX.P(), STDEVX.S(), VARX.P(), VARX.S()

Разница между версиями P и S точно такая же, как между функциями СТАНДОТКЛОН.Г() и СТАНДОТКЛОН.В() в обычном Excel. Версия P используется, когда набор данных представляет генеральную совокупность, а функция S, когда имеется только выборка данных. Это вопрос статистики, а не DAX (подробнее см. [СТАНДОТКЛОН.В и СТАНДОТКЛОН.Г: в чем различие?](http://baguzin.ru/wp/?p=15634)).

### COUNTX() и COUNTAX()

Вернемся к нашему примеру с SUMX(): SUMX(VALUES(Products[ModelName]); [Sales per Day]). Здесь имелось 119 уникальных значений ModelName, из которых только 3 имели непустые значения для [Sales per day]. Если бы мы заменили SUMX() на COUNTX(),мы бы получили 3, потому что COUNTX() не считает пустые значения. Таким образом, мы можем думать о COUNTX() как о "COUNT NONBLANKX()". Чем же тогда COUNTX() отличается от COUNTROWS()? Функция COUNTROWS() не может принимать меру в качестве аргумента, поэтому ее нельзя использовать для оценки того, сколько раз эта мера возвращает непустое значение.

COUNTAX() в отличие от COUNTX() считает значимыми пустые текстовые строки ="".

### Использование функций X для полей, которые не отображаются

Выше мы использовали SUMX() для того, чтобы суммировать итоги «правильно». Другое использование функций X связано с перебором полей, которые не выведены в сводной таблице. Чтобы проиллюстрировать это, добавим новую меру: [Max Single-Country Sales] = MAXX(VALUES(SalesTerritory[Country]); [Total Sales])



Рис. 16.5. Вы уверены, что понимаете, что именно отображает новая мера?

Добавим страну в сводную таблицу:



Рис. 16.6. Мера [Max Single-Country Sales] возвращает максимальные продажи в одной стране с учетом контекста фильтра

Значение в ячейке Е3 равно сумме значений в ячейках Е5, Е12 и Е19. Поскольку контекст фильтра в ячейке Е3: Year=2001, Products=All, то найдены максимальные продажи по стране по всем моделям. Посмотрите еще раз на рис. 16.5. Поле Country отсутствует в сводной таблице, но значения в столбце Е соответствуют только одной стране.

### RANKX()

Используем меру [Customer Sales Rank] = RANKX(ALL(Customers[FullName]); [Total Sales])



Рис. 16.7. Ранг по сумме продаж

Единственное различие, которое мы видим по сравнению с другими Х функциями заключается в том, что мы использовали ALL() вместо VALUES() в первом аргументе. Функция VALUES() вернет уникальное значение Customer[FullName] для каждого вхождения, поэтому мера RANKX ранжирует каждого клиента, как если бы он был единственным клиентом в мире. В тоже время ALL() ранжирует каждого клиента среди всех остальных.

Обратите внимание на нижнюю часть сводной таблицы:



Рис. 16.8. Если имеется несколько одинаковых значений (здесь это $4), следующий ранг отстоит на число таких повторений

Если вас это не устраивает, выход есть. На самом деле функция RANKX() имеет пять параметров (три последних – необязательные):



Рис. 16.9. Необязательные параметры функции RANKX()

Параметр [Значение] – возможно самый загадочный во всей Power Pivot. Мы рекомендуем всегда оставлять его пустым (мы рассмотрим его еще в главе 17). [Порядок] позволяет управлять порядком рангов: по убыванию, когда самому большому значению соответствует ранг 1 (значение параметра 0, DESC или пусто) или возрастанию (значение 1 или ASC). [Равные] говорит о том, как ранжировать одинаковые значения: пропустить число рангов равное числу повторений (0, Skip или пусто), расположить ранги плотно (1 или Dense):

=RANKX(ALL(Customers[FullName]); [Total Sales];;;Dense)



Рис. 16.10. «Плотное» расположение рангов в случае, если [Равные]=Dense

Если у вас есть два клиента с одним и тем же полным именем, они будут объединены в одного клиента и ранжированы несправедливо высоко. Поэтому убедитесь, что вы ранжируете по уникальному полю. Мы рекомендуем объединить CustomerKey или что-то уникальное с полным именем, чтобы вы могли распознать клиента по имени и сохранить уникальность.

Больше о ранжировании вы можете узнать в [нашей](https://powerpivotpro.com/2015/06/rankx-apalooza-within-and-across-groups-with-filters-etc/) заметке о ранжировании элементов внутри и между группами, с применением фильтров и т.п. (например, фруктовое мороженое может быть №1 в продаже мороженого, но быть №23 в десертах).

### TOPN()

TOPN() – не X-функция, но она очень близка к RANKX, так что будет рассмотрена здесь.

TOPN(<n\_value>, <table>, <orderBy\_expression>, [<order>],...)



Рис. 16.11. Синтаксис функции TOPN()

Функция возвращает N верхних строк указанной таблицы. RANKХ присваивает числовой ранг. TOPN ранжирует строки, а затем фильтрует их по N верхним строкам (N задано пользователем) и возвращает этот набор строк. Мы используем TOPN, когда нам нужно отфильтровать несколько верхних строк (скажем, самые популярные продукты, лучшие клиенты, лучшие продавцы и т.д.), а затем выполнить с ними вычисления. Обратимся к примеру. Ниже приведен объем продаж и количество проданных продуктов в 2002 г.:



Рис. 16.12. Объем продаж довольно стабилен от месяца к месяцу

Мы хотим посмотреть, насколько большой вклад в продажи вносят пять самых продаваемых продуктов (разных в разные месяцы). Создадим меру:

=CALCULATE(

[Total Sales];

TOPN(5; Products; [Total Sales])

)

Первый аргумент функции TOPN – 5 – говорит сколько верхних строк, отсортированных по рангу, должно быть возвращено. Второй аргумент – Products – говорит, из какой таблицы должны быть извлечены строки. (Вы не ограничены передачей имен таблиц, которые существуют в вашей модели данных. Можно использовать любое выражение DAX, возвращающее таблицу, например VALUES(), FILTER(), …) Третий аргумент – [Total Sales] – говорит, по каким данным определять пять верхних строк: по объему продаж, ценам прайс-листа, маржинальности… Здесь можно указать любую меру, подходящую для ранжирования строк.

Добавим еще одну *производную* меру, показывающую долю продаж пяти продуктов, дающих наибольший вклад: =DIVIDE([Sales for Top 5 Products]; [Total Sales]) (Напомню, что функция DIVIDE от простого деления отличается тем, что при делении на ноль DIVIDE вернет ноль, а простое деление – значение ошибки.)



Рис. 16.13. TOPN помогает наглядно увидеть, что происходит с нашим бизнесом

В первом полугодии топовые модели давали слишком большую долю продаж. Во втором полугодии удалось снизить зависимость от продажи топовых моделей.

### В качестве второго аргумента Х-функции не обязательно использовать меру

На самом деле вы можете нарушить правило «не использовать голые столбцы». Во втором аргументе Х-функции допустимо ввести имя столбца. Вы даже можете поместить туда формулу в стиле вычисляемого столбца, например, Customers[YearlyIncome]/Customers[NumberOfChildren].

### COUNTAX()

Когда вы используете меру в качестве второго аргумента, то рассчитываете, что функции COUNTX() и COUNTAX() вернут одинаковые результаты. Но COUNTAX() позволяет использовать в качестве второго аргумента также и текстовый столбец; тогда как COUNTX(), если вы используете столбец в качестве второго аргумента, требует, чтобы он был числовым или типом даты.

Для иллюстрации создадим небольшую табличку, которую мы добавили в окно Power Pivot:



Рис. 16.14. COUNTAX() позволяет использовать имя столбца в качестве второго аргумента

А вот и мера, использующая столбец в качестве второго аргумента: [COUNTAX Test] = COUNTAX(CountTest; CountTest[Column1]). Мера возвращает 7 из 8-строчной таблицы. Таким образом, мера не посчитала пустую строку. Измените COUNTAX() на COUNTX() и мы получим ошибку: COUNTX() отказывается принимать текстовый столбец в качестве второго аргумента.