PBIG
POWER BI GEBRUIKERSGROEP

Understanding CALCULATE – the queen of all DAX functions

Francesco Bergamaschi, M.Eng, M.Econ, MBA
Professor & Consultant
francesco.bergamaschi@kubisco.com

A big thank you to our amazing partners

Thanks

About me

- Master of Science in IT Engineering, Master of Science in Economics, MBA
- Professor of BI & Analytics @UniBO & @Bologna Business School
- Visiting Professor @UNIVPM e @UNICATT
- BI & Analytics Consultant
- Lecturer in DAX and Tabular since 2014
- Co-founding member of kubisco (www.kubisco.com)
- Co-founding member of the Power BI User Group Italy
- Current Focus: Composite Models and Visual Context

kubisco www.kubisco.com

Italy Power BI User Group

Outline

- Introducing CALCULATE
- Why modifying the Filter Context?
- What modifications can be done to the Filter Context through CALCULATE?
- Filter and Filter Context Definition
- More on Filters
- Filters vs Global Modifiers
- Filter Modifier
- Adding Filters *explicitly and overwriting* existing ones
- Removing Filters
- Modify the model Relationships Columns
- Modify the model Relationships cross-filter direction
- CALCULATE Global Modifiers
- CALCULATE Algorithm
- CALCULATE Alogrithm re-cap examples

Foreword

CALCULATE is preceived as a complex function. DAX is perceived as a complex language.

Both are misperceptions.

CALCULATE is a **simple** and **powerful** function. DAX is a **simple** and **powerful** language.

Point is that **simple** does not mean **easy**.

CALCULATE and DAX in general can be used in complex scenarios as they are powerful but you can also use both in simple scenarios and complexity does not show up

It is what we want to do that MIGHT be complex, not CALCULATE or DAX in general. As they are both powerful, theyt can ALSO be used in COMPLEX scenarios

Foreword

DAX Pillars are only six (learn them)!

Filter Context
Row Context
Iterators
Context Transition
Expanded Tables
Visual Context

Introducing CALCULATE

CALCULATE documentation (dax.guide)

CALCULATE (<Scalar Expression>, [Filter/GlobalModifier 1], [Filter/GlobalModifier 2], …)

In short, CALCULATE evaluates a **scalar** DAX expression in a Filter Context modified by Filters and/or Modifiers.

CALCULATE works ONLY with the Filter Context. The **scalar** expression must, therefore, be meaningful in absence of Row Context

Introducing CALCULATE

CALCULATE is implicitely called anytime a reference to a measure is done.

[Measure] is executed as CALCULATE ([Measure])

Due to the Context Transition (more on this later) performed by CALCULATE, the practice of referencing measures omitting the table name has been put in force ([Measure] and not Table[Measure]). In fact, Measures can be placed in any table without affecting their results. They have nothing to do with any particular table.

Columns, on the contrary, are hardwired to tables, so it makes sense to reference them as Table[Column]

Introducing CALCULATE

CALCULATE works ONLY with the Filter Context. The **scalar** expression must, therefore, **be meaningful in absence of Row Context**

CALCULATE ([Measure], …)	OK
CALCULATE (< Explicit scalar DAX code >, …)	OK
CALCULATE (Table[Column], ….)	NOT OK (Row Context needed)
CALCULATE (RELATED (Table[Column], …)	NOT OK (Row Context needed)

Introducing CALCULATE

CALCULATETABLE semantic is identical to CALCULATE, the only difference being:

CALCULATE evaluates a **scalar** DAX expression in a Filter Context modified by Filters and/or Global Modifiers,

while

CALCULATETABLE evaluates a **table** DAX expression in a Filter Context modified by Filters and/or Global Modifiers.

We shall, for brevity, describe CALCULATE only

Introducing CALCULATE

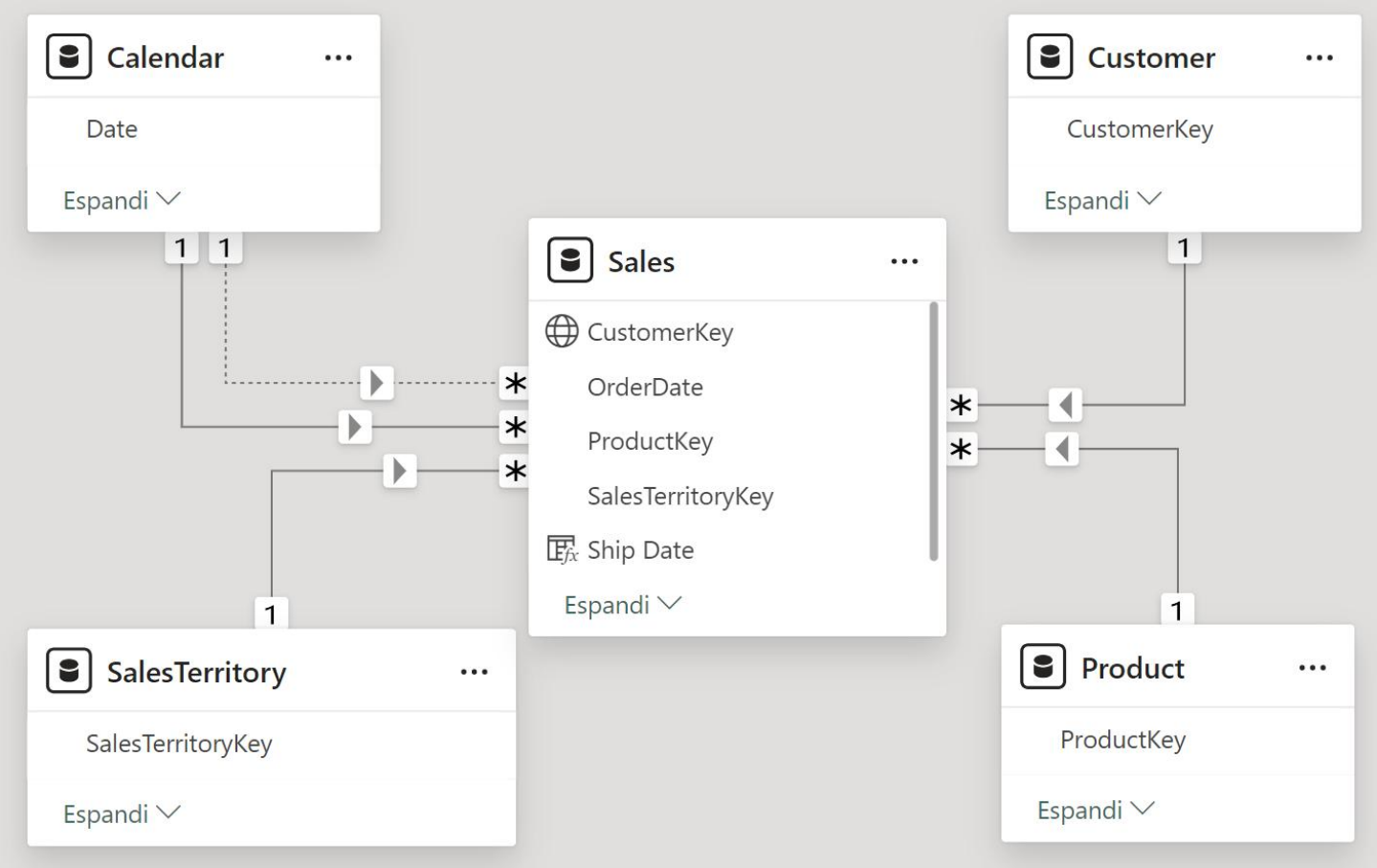
CALCULATE (<Scalar Expression>, [Filter/GlobalModifier 1], [Filter/GlobalModifier 2], …)

In short, CALCULATE evaluates a DAX expression in a Filter Context modified by Filters and/or Global Modifiers.

Why modifying the Filter Context? Many reasons:
1 – to avoid changing the internals of measures when creating variants;
2 – to avoid getting more than what is needed from the Filter Context and selecting what is needed *ex-post*;
3 – to simplify and shorten the DAX code;
4 – to simplify the DAX code maintenance

Introducing CALCULATE

Data model:



Why modifying the Filter Context?

```
1 Sales =
2 SUMX(
3     Sales,
4     Sales[OrderQuantity]*Sales[UnitPrice]
5 )
```

The purpose is, when a variation of a measure, in this case [Sales], is needed….

Color	Sales
Black	3.851.091
Blue	860.381
Multi	42.099
NA	184.354
Red	953.203
Silver	2.044.407
White	2.230
Yellow	1.853.296
Totale	**9.791.060**

Why modifying the Filter Context?

```
1 Sales M Customers =
2 SUMX(
3     FILTER (
4         Sales,
5         RELATED(Customer[Gender])="M"
6     ),
7     Sales[OrderQuantity]*Sales[UnitPrice]
8 )
```

Color	Sales	Sales M Customers
Black	3.851.091	1.917.049
Blue	860.381	403.857
Multi	42.099	21.310
NA	184.354	92.449
Red	953.203	511.608
Silver	2.044.407	974.592
White	2.230	1.142
Yellow	1.853.296	936.731
Totale	**9.791.060**	**4.858.738**

CalendarYear
- ☐ 2001
- ☐ 2002
- ■ 2003
- ☐ 2004

...To avoid doing this...

Why modifying the Filter Context?

```
1 Sales =
2 SUMX(
3     Sales,
4     Sales[OrderQuantity]*Sales[UnitPrice]
5 )
```

```
1 Sales M Customers =
2 SUMX(
3     FILTER (
4         Sales,
5         RELATED(Customer[Gender])="M"
6     ),
7     Sales[OrderQuantity]*Sales[UnitPrice]
8 )
```

Color	Sales	Sales M Customers
Black	3.851.091	1.917.049
Blue	860.381	403.857
Multi	42.099	21.310
NA	184.354	92.449
Red	953.203	511.608
Silver	2.044.407	974.592
White	2.230	1.142
Yellow	1.853.296	936.731
Totale	**9.791.060**	**4.858.738**

CalendarYear
- ☐ 2001
- ☐ 2002
- ■ 2003
- ☐ 2004

Here no change to the Filter Context is applied, so:
1 – We need to change the internals of the measure;
2 – We are getting more than what we need from the Filter Context (“M” and “F”) and selecting what is needed (only “M”) *ex-post*;
3 – the DAX code is pretty long and not so easy to read and understand

Why modifying the Filter Context?

```
1 Sales M Customers CALCULATE =
2 CALCULATE(
3     [Sales],
4     Customer[Gender] = "M"
5 )
```

Color	Sales	Sales M Customers	Sales M Customers CALCULATE
Black	3.851.091	1.917.049	1.917.049
Blue	860.381	403.857	403.857
Multi	42.099	21.310	21.310
NA	184.354	92.449	92.449
Red	953.203	511.608	511.608
Silver	2.044.407	974.592	974.592
White	2.230	1.142	1.142
Yellow	1.853.296	936.731	936.731
Totale	**9.791.060**	**4.858.738**	**4.858.738**

CalendarYear
- ☐ 2001
- ☐ 2002
- ■ 2003
- ☐ 2004

And do this!

Why modifying the Filter Context?

```
1 Sales =
2 SUMX(
3     Sales,
4     Sales[OrderQuantity]*Sales[UnitPrice]
5 )
```

Color	Sales	Sales M Customers CALCULATE
Black	3.851.091	1.917.049
Blue	860.381	403.857
Multi	42.099	21.310
NA	184.354	92.449
Red	953.203	511.608
Silver	2.044.407	974.592
White	2.230	1.142
Yellow	1.853.296	936.731
Totale	**9.791.060**	**4.858.738**

CalendarYear
- 2001
- 2002
- 2003
- 2004

```
1 Sales M Customers CALCULATE =
2 CALCULATE(
3     [Sales],
4     Customer[Gender] = "M"
5 )
```

Here a Filter on Customer[Gender] has been applied to the Filter Context, so:

1 – We are changing the Filter Context, so we do NOT need to change the measure internals and we can reference it;

2 – We are, therefore, getting only what we need from the Filter Context (only “M”);

3 – the DAX code is short and easy to read, understand and maintain

What modifications can be done to the Filter Context through CALCULATE?

CALCULATE (<Scalar Expression>, [Filter/GlobalModifier 1], [Filter/GlobalModifier 2], …)

1 – Adding Filters (explicitly/implicitly, overwriting/intersecting existing ones);

2 – Removing Filters;

3 – Modify the Columns involved in the model Relationships;

4 – Modify the Cross-Filter direction in the model Relationships

What modifications can be done to the Filter Context through CALCULATE?

CALCULATE (<Scalar Expression>, [Filter/GlobalModifier 1], [Filter/GlobalModifier 2], …)

CALCULATE Filters are applied to the Filter Context in a logical AND

Example: CALCULATE ([Sales], Filter 1, Filter 2, …, Filter N)

Filter 1, Filter 2, … Filter N will be applied in AND (they must all be valid at the same time).

Therefore, AND conditions are natural and easy, while OR conditions are somehow challenging in CALCULATE

Filter and Filter Context Definition

CALCULATE (<Scalar Expression>, [Filter/GlobalModifier 1], [Filter/GlobalModifier 2], …)

A Filter is a set of tuples for one or more columns.

A set of Filters is called Filter Context.

CALCULATE Filters can be expressed, **in some circumstances**, as Predicates.

Example: CALCULATE ([Sales], Products[Color] = "Red")

More on Filters

CALCULATE (<Scalar Expression>, [Filter/GlobalModifier 1], [Filter/GlobalModifier 2], …)

CALCULATE Filters, though, are Tables, not Predicates.

The syntax sugar of a predicate can be used when filtering a set of columns from the same table for a set of specific values, that can also be expressed as a DAX expression (NOT an explicit call to a measure, though, more on this later).

Examples ALLOWED:

```
CALCULATE ( [Sales], Products[Color] = "Red" )
CALCULATE ( [Sales], Products[Color] = "Red", Products[Size] = "S" )
CALCULATE ( [Sales], Products[Color] = "Red" && Products[Size] = "S" )
CALCULATE ( [Sales], Products[Color] = "Red" || Products[Size] = "S" )
CALCULATE ( [Sales], Products[ListPrice] = MAX ( Products[ListPrice] )
```

More on Filters

CALCULATE (<Scalar Expression>, [Filter/GlobalModifier 1], [Filter/GlobalModifier 2], …)

CALCULATE Filters, though, are Tables, not Predicates.

This means it is utterly important to be expert of DAX table functions like

SUMMARIZE
CROSSJOIN
GENERATE
VALUES
DISTINCT
FILTER
ALL XXX

To create the filter you need !

More on Filters

CALCULATE (<Scalar Expression>, [Filter/GlobalModifier 1], [Filter/GlobalModifier 2], …)

CALCULATE Filters, though, are Tables, not Predicates.

The syntax sugar of a predicate cannot be used when filtering a set of columns from different tables for a set of specific values and when the values are to be calculated through a measure.

Examples NOT ALLOWED:

1 CALCULATE ([Sales], Products[Color] = [Top Sales Color])
2 CALCULATE ([Sales], Products[Color] = "Red" || Customer[EnglishEduction] = "Bachelors")
3 CALCULATE ([Sales], Products[Color] = "Red" && Customer[EnglishEduction] = "Bachelors")

note: the 3° example can be solved simply by
CALCULATE ([Sales], Products[Color] = "Red«, Customer[EnglishEduction] = "Bachelors")

More on Filters

CALCULATE (<Scalar Expression>, [Filter/GlobalModifier 1], [Filter/GlobalModifier 2], …)

CALCULATE Filters are Tables, not Predicates.

The real DAX code executed when you write a predicate is the following:

Example:

```
CALCULATE ( [Sales], Products[Color] = "Red" )
```

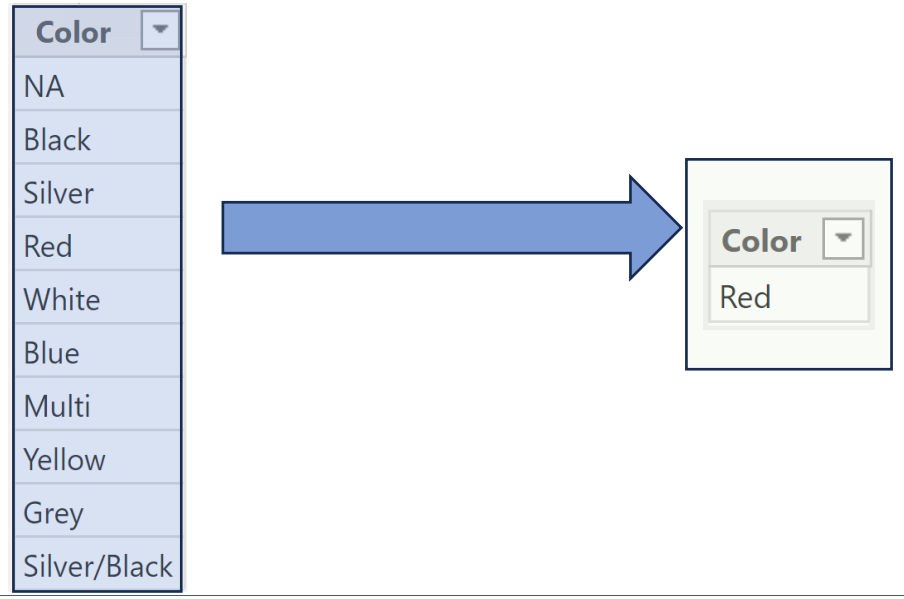
is translated into:

```
CALCULATE ( [Sales], FILTER ( ALL ( Products[Color] ), Products[Color] = "Red" ) )
```

More on Filters

CALCULATE ([Sales], Products[Color] = "Red") is translated into:

CALCULATE ([Sales], FILTER (ALL (Products[Color]), Products[Color] = "Red"))



More on Filters

CALCULATE ([Sales], Products[Color] = "Red" || Products[Size] = "M") is translated into:

```
CALCULATE (
        [Sales],
        FILTER (
                ALL ( Products[Color], Products[Size] ),
                Products[Color] = "Red" || Products[Size] = "M"
        )
)
```

Color	Size
NA	
Black	
Silver	
Red	
Blue	
Multi	
Grey	
Silver/Black	
Black	58
Red	58
Blue	58
Yellow	58
Black	M
White	M
Blue	M
Multi	M
Yellow	M
• • • •	

Color	Size
Red	
Red	58
Black	M
White	M
Blue	M
Multi	M
Yellow	M
Red	62
Red	44
Red	48
Red	52
Red	56
Red	60

More on Filters

CALCULATE ([Sales], Products[Color] = "Yellow" && Products[Size] = "M") is translated into:

```
CALCULATE (
        [Sales],
        FILTER (
                ALL ( Products[Color], Products[Size] ),
                Products[Color] = "Yellow" && Products[Size] = "M"
        )
)
```

Color	Size
NA	
Black	
Silver	
Red	
Blue	
Multi	
Grey	
Silver/Black	
Black	58
Red	58
Blue	58
Yellow	58
Black	M
White	M
Blue	M
Multi	M
Yellow	M
....	

Color	Size
Yellow	M

OR between columns of differente tables

CALCULATE ([Sales], Products[Color] = "Red" || Customer[EnglishEduction] = "Bachelors")

is not an allowed syntax , but you can solve this creating your own explicit table filter, a first

(not nice) attempt might be

```
Sales Red Products OR Bachelors Customers =
CALCULATE (
    [Sales],
    FILTER (
        Sales,
        RELATED ( 'Product'[Color] ) = "Red" ||
        RELATED( Customer[EnglishEducation] ) = "Bachelors"
    )
)
```

OR between columns of differente tables

```
Sales Red Products OR Bachelors Customers =
CALCULATE (
    [Sales],
    FILTER (
        Sales,
        RELATED ( 'Product'[Color] ) = "Red" ||
        RELATED( Customer[EnglishEducation] ) = "Bachelors"
    )
)
```

SalesTerritory Group	Sales	Sales Red Products	Sales Red Products OR Bachelors Customers
Europe	3.382.979	400.146	1.305.743
North America	3.374.297	168.235	1.178.300
Pacific	3.033.784	384.821	1.639.090
Total	**9.791.060**	**953.203**	**4.123.133**

More on Filters

FILTER COLUMNS AND NOT TABLES PLEASE, the below code will inject the entire set of columns of the Products tabel into the Filter Context when you only need two columns for the filter you are building

```
CALCULATE (
        [Sales],
        FILTER (
                Products,
                Products[Color] = "Yellow" && Products[Size] = "M"
        )
)
```

More on Filters

FILTER COLUMNS AND NOT TABLES PLEASE, **but what about when you** cannot, like (it seems) in the case of OR between columns of different tables ? **But** *are we sure we really cannot do better than injecting the ENTIRE expanded Sales table in the filter context? We only need 2 columns to create the filter!*

```
Sales Red Products OR Bachelors Customers =
CALCULATE (
    [Sales],
    FILTER (
        Sales,
        RELATED ( 'Product'[Color] ) = "Red" ||
        RELATED( Customer[EnglishEducation] ) = "Bachelors"
    )
)
```

More on Filters

YES, we can do better thanks to table functions (only two columns):

```
Sales Red Products OR Bachelors Customers IMPROVED DAX =
CALCULATE (
    [Sales],
    FILTER (
        SUMMARIZE( Sales, 'Product'[Color], Customer[EnglishEducation] ),
        'Product'[Color] = "Red" ||
        Customer[EnglishEducation] = "Bachelors"
    )
)
```

SalesTerritory Group	Sales	Sales Red Products	Sales Red Products OR Bachelors Customers	Sales Red Products OR Bachelors Customers IMPROVED DAX
Europe	3.382.979	7.724.331	1.305.743	1.305.743
NA		7.724.331		
North America	3.374.297	7.724.331	1.178.300	1.178.300
Pacific	3.033.784	7.724.331	1.639.090	1.639.090
Total	**9.791.060**	**7.724.331**	**4.123.133**	**4.123.133**

CalendarYear

- ☐ 2001
- ☐ 2002
- ■ 2003
- ☐ 2004

Filters vs Global Modifiers

CALCULATE (<Scalar Expression>, [Filter/GlobalModifier 1], [Filter/GlobalModifier 2], …)

The second and subsequent inputs can either be a Filter or a Global Modifer. Global Modifiers are not Filters, they are instructions to temporarily change the model:

(i) Removing Filters;

(ii) Modifying the Columns involved in the model Relationships;

(iii) Modifying the Cross-Filter direction in the model Relationships

Filter Modifier

CALCULATE (<Scalar Expression>, [Filter/GlobalModifier 1], [Filter/GlobalModifier 2], …)

Each Filter can be applied with or without a Filter Modifier called KEEPFILTERS, see next slides for details. A Filter Modifer is not a Global Modifier as it affects only the semantics of a specific Filter

1 – Adding Filters *explicitly and overwriting* existing ones

```
1 Sales M Customers CALCULATE =
2 CALCULATE(
3     [Sales],
4     Customer[Gender] = "M"
5 )
```

Color	Sales	Sales M Customers	Sales M Customers CALCULATE
Black	3.851.091	1.917.049	1.917.049
Blue	860.381	403.857	403.857
Multi	42.099	21.310	21.310
NA	184.354	92.449	92.449
Red	953.203	511.608	511.608
Silver	2.044.407	974.592	974.592
White	2.230	1.142	1.142
Yellow	1.853.296	936.731	936.731
Totale	**9.791.060**	**4.858.738**	**4.858.738**

CalendarYear
- ☐ 2001
- ☐ 2002
- ■ 2003
- ☐ 2004

Filter Context in which [Sales], within CALCULATE, is executed

Filter Context in which [Sales M Customers CALCULATE] is executed

Color	CalendarYear
Red	2003

Gender
M

1 – Adding Filters *explicitly and overwriting* existing ones

```
1 Sales M Customers CALCULATE =
2 CALCULATE(
3     [Sales],
4     Customer[Gender] = "M"
5 )
```

Gender	Sales	Sales M Customers	Sales M Customers CALCULATE
F	4.932.322		4.858.738
M	4.858.738	4.858.738	4.858.738
Totale	**9.791.060**	**4.858.738**	**4.858.738**

CalendarYear
- ☐ 2001
- ☐ 2002
- ■ 2003
- ☐ 2004

Filter Context in which [Sales], within CALCULATE, is executed

Filter Context in which [Sales M Customers CALCULATE] is executed

Ge	CalendarYear
F	2003

Gender
M

1 – Adding Filters *explicitly and overwriting* existing ones

The explicitely added Filters will be applied with OVERWRITE policy (eliminate existing Filters on Columns when adding the new ones). To turn the policy to INTERSECT (keeping existing Filters on columns when adding the new ones) you need to use the KEEPFILTERS Filter Modifier on the Filter call.

Here we have shown examples of the OVERWRITE approach, now let us go for the INTERSECT approach

1 – Adding Filters *explicitly and intersecting* existing ones

```
1 Sales M Customers CALCULATE Intersect =
2 CALCULATE(
3     [Sales],
4     KEEPFILTERS(Customer[Gender] = "M")
5 )
```

Gender	Sales	Sales M Customers	Sales M Customers CALCULATE Intersect
F	4.932.322		
M	4.858.738	4.858.738	4.858.738
Totale	**9.791.060**	**4.858.738**	**4.858.738**

CalendarYear
- ☐ 2001
- ☐ 2002
- ■ 2003
- ☐ 2004

Filter Context in which [Sales], within CALCULATE, is executed

Filter Context in which [Sales M Customers CALCULATE] is executed

Gender	CalendarYear
F	2003

Gender
M

1 – Adding Filters *implicitely and overwriting* existing ones

CALCULATE performs Context Transition on all row contexts that are active at the time of the call and on **expanded version tables (so please iterate only the minimum number of columsn you need!)**. The Filters injected in the Filter Context through this process will be applied with OVERWRITE policy (to turn the policy to INTERSECT you need to use the Filter Modifier KEEPFILTERS on the iterated table, this is not very common and needed basically when there is an arbitrarily-shaped set to deal with)

1 – Adding Filters *implicitely and overwriting* existing ones

Context Transition consists in the invalidation of any Row Context and the creation of an **equivalent** Filter Context.

In detail, this is what Context Transition does:

- it injects in the Filter Context a set of Fiters - for each column of all the row contexts that are active at the time of the call to CALCULATE, a Filter is placed with OVERWRITE for the value of that column. Inner Row Contexts will prevail on outer;

- it invalidates all Row Contexts active at the time of the call.

1 – Adding Filters *implicitely and overwriting* existing ones

Context Transition **does not guarantee** to isolate a single row (the currently iterated one), that is why it creates and **equivalent** Filter Context.

Therefore, Context Transition shoud never be triggered on tables that do not have a primary key

The following two examples show Context Transition on a Calculated Column and in a Measure (the OVERWRITE/INTERSECT part of the Filter placing is irrelevant in these two examples, later on a specific example will be shown)

1 – Adding Filters *implicitely and overwriting* existing ones

Context Transition in a Calculated Column:

OVERWRITE / INTERSECT is irrelevant as the Filter Context is initially empty

Filter Context after Context Transition

PK	EPN	COLOR	SSL
313	Road...	Red	100

LP	S	CLASS
3578,27	52	H

```
1 Product Sales Column =
2 [Sales]
```

ProductKey	EnglishProductName	Color	SafetyStockLevel	ListPrice	Size	Class	Product Sales Column
312	Road-150 Red, 48	Red	100	3578,27	48	H	1.205.877
310	Road-150 Red, 62	Red	100	3578,27	62	H	1.202.299
313	Road-150 Red, 52	Red	100	3578,27	52	H	1.080.638
314	Road-150 Red, 56	Red	100	3578,27	56	H	1.055.590
							1.005.494
							979.961
							979.036
							961.601
							954.716

The Filter Context is initially empty (this is a calculated Column) and a Row Context is active on he Product Table. The implicit CALCULATE of a measure reference triggers Context Transition. The Filter Context is then filled with Filters and the Row Context is invalidated. In case of multiple row contexts, this will happen for all of them. The inner Row Context will prevail on the outer.

1 – Adding Filters *implicitely and overwriting* existing ones

```
1  Sales Cust Over 1.000 =
2  SUMX(
3      VALUES( Customer[CustomerKey] ),
4      VAR SalesCurrentCustomerCurrentContext = [Sales]
5      RETURN
6      IF (
7          SalesCurrentCustomerCurrentContext > 1000,
8          SalesCurrentCustomerCurrentContext
9      )
10 )
```

CY
2001

CustKey
11055

Customerkey	[Sales]
11000	34.000
11055	68.000
12078	45.000
13456	57.000
13689	13.000

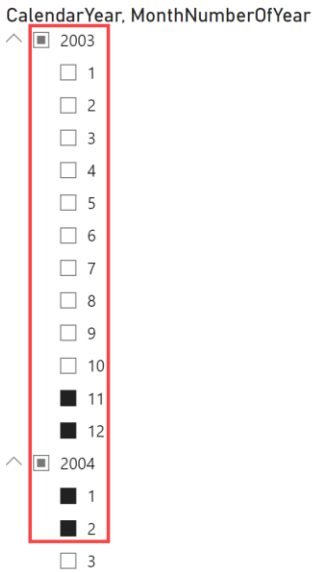
Context Transition in a Measure: here the *Sales Cust Over 1.000* Measure will only consider Customers that, in the current selection, have Sales above 1.000. OVERWRITE is again irrelevant as there is no Filter, initally, on the Customer table in the Filter Context

CalendarYear	Sales	Sales Cust Over 1.000
2001	3.266.374	3.187.376
2002	6.530.344	6.228.613
2003	9.791.060	8.727.456
2004	9.770.900	8.321.146
Total	**29.358.677**	**27.847.299**

1 – Adding Filters *implicitely and intersecting* existing ones

```
1 Monthly Average Sales =
2 AVERAGEX (
3     KEEPFILTERS( VALUES ( 'Calendar'[MonthNumberOfYear] ) ),
4     [Sales]
5 )
```

CalendarYear	Sales	Monthly Average Sales
2003	**2.928.769**	**1.464.384**
11	1.196.981	1.196.981
12	1.731.788	1.731.788
2004	**2.802.725**	**1.401.362**
1	1.340.245	1.340.245
2	1.462.480	1.462.480
Totale	**5.731.494**	**1.432.873**



Filter Context in which [Monthly Average Sales] is executed

Year	Month
2003	11
2003	12
2004	1
2004	2

Average = 1.432.873

Month	Sales
11	**1.196.981**
12	**1.731.788**
1	**1.340.245**
2	**1.462.480**

Filter Context in which [Sales], due to the implicit CALCULATE and the Context Transition, is executed, Row by row on VALUES (Calendar[Month NumberOfYear])

1 – Adding Filters *implicitely and intersecting* existing ones

PBIG
POWER BI GEBRUIKERSGROEP

Note:

In the preceding example, simply iterating on VALUES (Calendar[YearMonth]), which uniquely idetifies each month in the arbitrarily-shaped set, would solve and no KEEPFILTERS would be needed. Still, the preceding (simple ?!) example explains the issue, hopefully. Real-world measures in which you have no other choice than using KEEPFILTERS on the iterated table are very complex and would not fit this session. Example: Measures for budget allocations. To get some examples, visit www.daxpatterns.com

2 – Removing Filters

```
1 ALL Sales CALCULATE =
2 CALCULATE(
3     [Sales],
4     REMOVEFILTERS( )
5 )
```

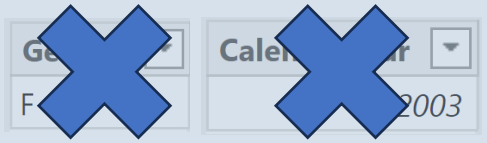
Gender	Sales	ALL Sales CALCULATE
F	4.932.322	29.358.677
M	4.858.738	29.358.677
Totale	**9.791.060**	**29.358.677**

CalendarYear

- ☐ 2001
- ☐ 2002
- ■ 2003
- ☐ 2004

Filter Context in which [Sales], within CALCULATE, is executed

Filter Context in which [ALL Sales CALCULATE] is executed



2 – Removing Filters

REMOVEFILTERS is a Global Modifier and is an alias for ALL which, when used as a CALCULATE Global Modifier, does not act as a table function but, instead, removes filters. To avoid confusion, the REMOVEFILTERS alias was introduced a few years ago. As ALL, REMOVEFILTERS can be used with no arguments, with one entire table or with a set of columns from a single table. Note: REMOVEFILTERS cannot act as a table function, while ALL can

2 – Removing Filters

List of CALCULATE Global Modifiers to remove filters:

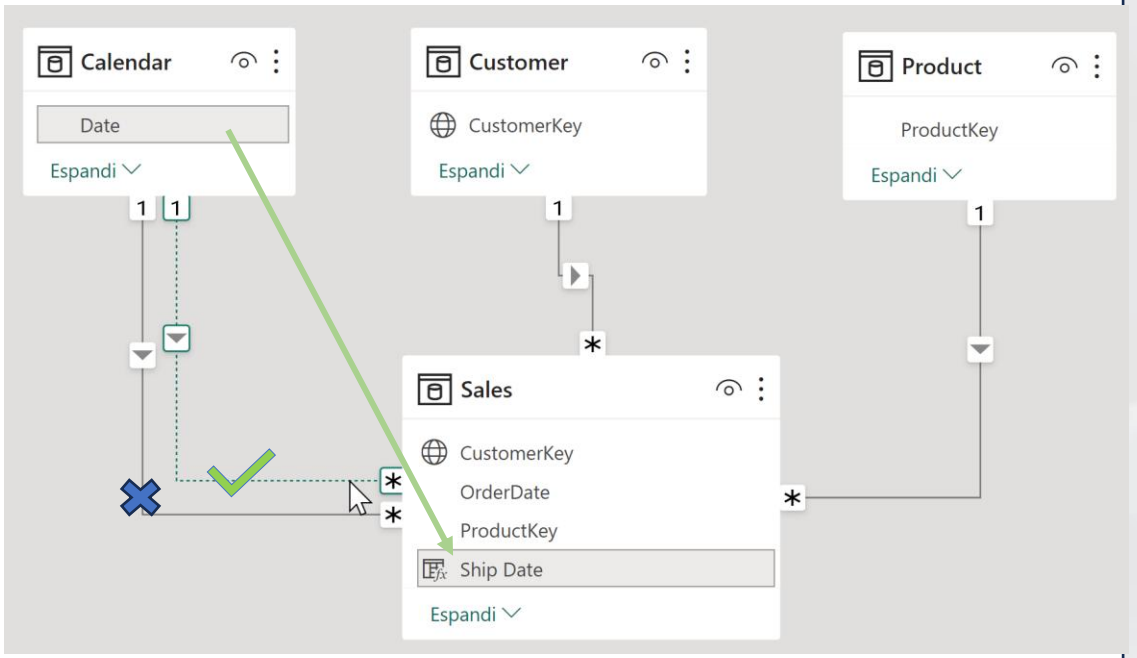
- REMOVEFILTERS ()
- ALL ()
- ALLSELECTED ()
- ALLEXCEPT ()

…

Any ALLXXX () function in other words! No time today to go through all of them ☹

3 – Modify the model Relationships Columns

```
1 Shippings =
2 CALCULATE(
3     [Sales],
4     USERELATIONSHIP( Sales[Ship Date],'Calendar'[Date] )
5 )
```

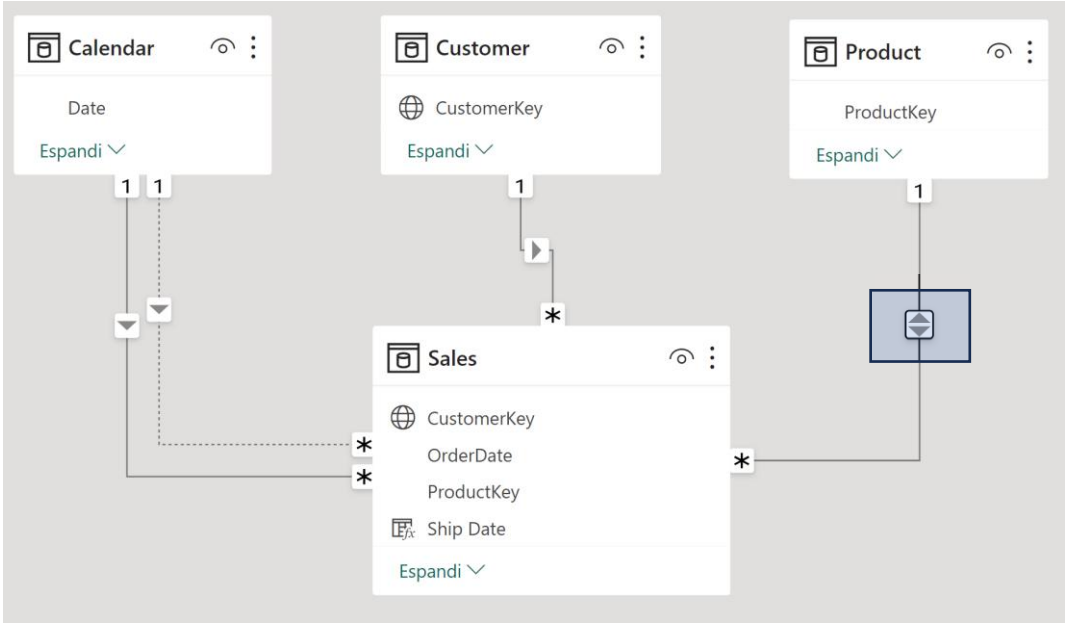


CalendarYear	Sales	Shippings
2001	3.266.374	3.105.587
2002	6.530.344	6.576.979
2003	9.791.060	9.517.549
2004	9.770.900	10.158.562
Totale	**29.358.677**	**29.358.677**

USERELATIONSHIP, a Global Modifier, will change the columns involved in the Relationships between the Calendar and Sales tables, so that Calendar[Date] will temporarily filter Sales[Ship Date] and not Sales[Order Date]. Once this is done, CALCULATE will evaluate [Sales]

4 – Modify the model Relationships cross-filter direction

```
1 # Colors Sold =
2 CALCULATE(
3     DISTINCTCOUNT( 'Product'[Color] ),
4     CROSSFILTER( Sales[ProductKey], 'Product'[ProductKey], BOTH )
5 )
```



CalendarYear Sales	# Colors Sold
2001	3.266.374	3
2002	6.530.344	4
2003	9.791.060	8
2004	9.770.900	8
Totale	**29.358.677**	**10**

CROSSFILTER, a Global Modifier, will change the Cross-Filter direction in the Relationships between the Product and Sales tables, so that it will temporarily become bi-directional, here the full options list:

- Both
- None
- OneWay
- OneWay_LeftFiltersRight
- OneWay_RightFiltersLeft

Only for many-to-many-cardinality relationships!

CALCULATE Global Modifiers

List of CALCULATE Global Modifiers:

- REMOVEFILTERS ()
- Any ALLXXX () function (they will NOT act as table functions when used as a top level function in CALCULATE)
- USERELATIONSHIPS ()
- CROSSFILTER ()

All these functions can be applied to the Filter Context (REMOVEFILTERS and ALLXXX) and to the model (USERELATIONSHIPS and CROSSFILTER) in any order, the effect will be the same

CALCULATE Algorithm

1 – Evaluate the explicit Filters, if any, **in the Filter Context active at the time of the call to CALCULATE** and create a copy of this Filter Context;

2 – Perform Context Transition on **all row contexts (RC) active at the time of the CALCULATE** call (the Filter Context starts to change, inner RC will prevail on outer);

3 – Apply the Global Modifiers, if any (further Filter Context change);

4 – Apply the explicit Filters, evaluated in step 1, if any (final Filter Context change), each **with our without the Filter Modifier KEEPFILTERS**;

5 – Evaluate the scalar expression in the modified Filter Context, return the result, then put the Filter Context active at the time of the call to CALCULATE back in force

CALCULATE Algorithm

The algorithm steps are performed in the specific order outlined, therefore the two following expressions give the same result:

CALCULATE ([Revenues], Products[Color] = "Red", REMOVEFILTERS (Products))

CALCULATE ([Revenues], REMOVEFILTERS (Products), Products[Color] = "Red")

In general, the order in which you insert Filters and Global Modifiers is irrelevant. What IS relevant is managing the calls to functions to take advantage of the algorithm

CALCULATE Algorithm

Focus on the details of the algorithm that are crucial to get the result we are looking for:

1 – Evaluate the explicit Filters, if any, **in the Filter Context active at the time of the call to CALCULATE** and create a copy of this Filter Context

The bold part is extremely important: Filters are a MEMORY to what the Filter Context was at the time of the CALCULATE call, so you can restore a part of it if you need to

CALCULATE Algorithm

Filters are a MEMORY to what the Filter Context was at the time of the CALCULATE call, so you can restore a part of it if you need to. Example:

```
1 Sales of the Year =
2 CALCULATE(
3     [Sales],
4     REMOVEFILTERS ( 'Calendar' ),
5     VALUES ( 'Calendar'[CalendarYear] )
6 )
```

CalendarYear EnglishMonth Name	2001 Sales	Sales of the Year	2002 Sales	Sales of the Year	2003 Sales	Sales of the Year	2004 Sales	Sales of the Year	**Total Sales**	**Sales of the Year**
January		3.266.374	596.747	6.530.344	438.865	9.791.060	1.340.245	9.770.900	**2.375.857**	**29.358.677**
February		3.266.374	550.817	6.530.344	489.090	9.791.060	1.462.480	9.770.900	**2.502.387**	**29.358.677**
March		3.266.374	644.135	6.530.344	485.575	9.791.060	1.480.905	9.770.900	**2.610.615**	**29.358.677**
April		3.266.374	663.692	6.530.344	506.399	9.791.060	1.608.751	9.770.900	**2.778.842**	**29.358.677**
May		3.266.374	673.556	6.530.344	562.773	9.791.060	1.878.318	9.770.900	**3.114.646**	**29.358.677**
June		3.266.374	676.764	6.530.344	554.799	9.791.060	1.949.361	9.770.900	**3.180.924**	**29.358.677**
July	473.388	3.266.374	500.365	6.530.344	886.669	9.791.060	50.841	9.770.900	**1.911.263**	**29.358.677**
August	506.192	3.266.374	546.001	6.530.344	847.414	9.791.060		9.770.900	**1.899.607**	**29.358.677**
September	473.943	3.266.374	350.467	6.530.344	1.010.258	9.791.060		9.770.900	**1.834.668**	**29.358.677**
October	513.329	3.266.374	415.390	6.530.344	1.080.450	9.791.060		9.770.900	**2.009.169**	**29.358.677**
November	543.993	3.266.374	335.095	6.530.344	1.196.981	9.791.060		9.770.900	**2.076.070**	**29.358.677**
December	755.528	3.266.374	577.314	6.530.344	1.731.788	9.791.060		9.770.900	**3.064.630**	**29.358.677**
Total	**3.266.374**	**3.266.374**	**6.530.344**	**6.530.344**	**9.791.060**	**9.791.060**	**9.770.900**	**9.770.900**	29.358.677	29.358.677

This Filter is evaluated in the Filter Context active at the time of the CALCULATE call, so before REMOVEFILTERS is called. Therefore, even though, after REMOVEFILTERS, there is no trace of the [CalendarYear] value, we can restore it!

CALCULATE Algorithm

Focus on the details of the algorithm that are crucial to get the result we are looking for:

2 – Perform Context Transition on **all row contexts (RC) active at the time of the CALCULATE** call (the Filter Context starts to change, inner RC will prevail on outer)

The bold part is again extremely important: If more than one (nested) row context was active at the time of the CALCULATE call, all of them will be converted to an equivalent Filter Context. Inner row contexts will prevail on outer

CALCULATE Algorithm

If more than one (nested) row context was active at the time of the CALCULATE call, all of them will be converted to an equivalent Filter Context. Inner row contexts will prevail on outer.

Example:

```
1 Nested Row Context Product Sales =
2 -- Outer Row Context on Table Product
3 SUMX(
4     -- Inner Row Context on Table Product
5     'Product',
6     [Sales]
7 )
```

A set of filters will be added to the initially empty Filter Context, one for each column in the outer Row Context

Here the inner Row Context will go again one row at a time scanning the full table, and the iterated row will prevail on the one iterated from the outer Row Context

CALCULATE will trigger Context Transition and invalidate both Row Contexts

EnglishProductName	Color	SafetyStockLevel	ListPrice	Size	Class	Product Sales Column	Nested Row Context Product Sales
Road-150 Red, 48	Red	100	3578,27	48	H	1.205.877	29.358.677
Road-150 Red, 62	Red	100	3578,27	62	H	1.202.299	29.358.677
Road-150 Red, 52	Red	100	3578,27	52	H	1.080.638	29.358.677
Road-150 Red, 56	Red	100	3578,27	56	H	1.055.590	29.358.677
Road-150 Red, 44	Red	100	3578,27	44	H	1.005.494	29.358.677

CALCULATE Algorithm

Focus on the details of the algorithm that are crucial to get the result we are looking for:

3 – Apply the Global Modifiers, if any (further Filter Context change)

The important detail here is that this step (step 3) comes after Context Transition (step 2): Global Modifers can, therefore, override Context Transition

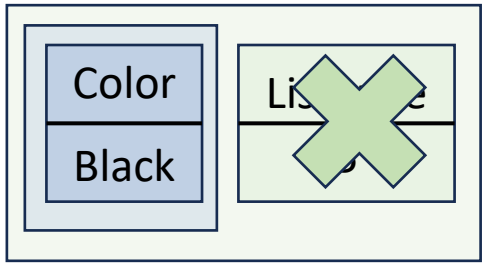
CALCULATE Algorithm

The important detail here is that this step (step 3) comes after Context Transition (step 2): Global Modifers can, therefore, override Context Transition. Example:

```
1 Sales MaxListPrice Products =
2 CALCULATE(
3     [Sales],
4     FILTER(
5         ALL ( 'Product'[ListPrice] ),
6         'Product'[ListPrice] =
7         CALCULATE( [Max List Price], REMOVEFILTERS( 'Product'[ListPrice] )
8         )
9     )
10 )
```

```
Max List Price = MAX ( 'Product'[ListPrice] )
```

Color	Sales	Sales MaxListPrice Products
Black	8.838.412	712.123
Blue	2.279.096	1.511.500
Multi	106.471	
NA	435.117	
Red	7.724.331	5.549.897
Silver	5.113.389	628.998
White	5.106	
Yellow	4.856.756	1.480.507
Totale	**29.358.677**	**5.549.897**



ALL (Product[ListPrice])	[MaxListPrice]	CALCULATE ([MaxListPrice], REMOVEFILTERS (Product[Listprice]))
5	5	15
7	7	15
9	9	15
12	12	15
15	15	15

CALCULATE Algorithm

The important detail here is that this step (step 3) comes after Context Transition (step 2): Global Modifers can, therefore, override Context Transition. Example with better DAX code, no Context Transition at all!:

```
1 Sales MaxListPrice Products IMPROVED DAX =
2 VAR _MaxListPrice = [Max List Price]
3 RETURN
4 CALCULATE(
5     [Sales],
6     'Product'[ListPrice] = _MaxListPrice
7 )
```

Color	Sales	Sales MaxListPrice Products	Sales MaxListPrice Products IMPROVED DAX
Black	8.838.412	712.123	712.123
Blue	2.279.096	1.511.500	1.511.500
Multi	106.471		
NA	435.117		
Red	7.724.331	5.549.897	5.549.897
Silver	5.113.389	628.998	628.998
White	5.106		
Yellow	4.856.756	1.480.507	1.480.507
Totale	**29.358.677**	**5.549.897**	**5.549.897**

Remember that Variables in DAX are… constant values!

CALCULATE Algorithm

Focus on the details of the algorithm that are crucial to get the result we are looking for:

4 – Apply the explicit Filters, evaluated in step 1 if any, to the Filter Context (final change), **with our without the Filter Modifier KEEPFILTERS**

The bold part is again extremely important: KEEPFILTERS is applied to a Filter, so it is always applied after Global Modifiers and you are free to decide, on each Filter, if you want or not KEEPFILTERS (INTERSECT)

CALCULATE Algorithm

When and why is KEEPFILTERS useful?

```
1 Trendy Color Sales =
2 CALCULATE(
3     [Sales],
4     'Product'[Color] IN {"Black", "Blue", "Silver"}
5 )
```

SalesTerritoryCountry	Sales	Trendy Color Sales
Australia	4.697.632	4.916.822
Canada	1.169.062	978.286
France	1.298.137	1.534.561
Germany	1.476.614	1.684.235
United Kingdom	1.681.648	2.031.972
United States	5.083.784	5.085.021
Total	**15.406.876**	**16.230.897**

How can Trendy Color Sales be higher than Sales? This is due to OVERWRITE. The existing Filter on the Color (slicer) will be removed before injecting the CALCULATE one and, therefore, always and only the three trendy colors will be considered

CALCULATE Algorithm

When and why is KEEPFILTERS useful (continued)?

```
1 Trendy Color Sales =
2 CALCULATE(
3     [Sales],
4     KEEPFILTERS(
5         'Product'[Color] IN {"Black", "Blue", "Silver"}
6     )
7 )
```

SalesTerritoryCountry	Sales	Trendy Color Sales
Australia	4.697.632	553.453
Canada	1.169.062	169.503
France	1.298.137	188.680
Germany	1.476.614	266.537
United Kingdom	1.681.648	321.907
United States	5.083.784	779.015
Total	**15.406.876**	**2.279.096**

Using KEEPFILTERS, the existing Filter on the Color (slicer) will be kept and then intersected with the Filter injected by CALCULATE

Note: you can decide on each Filter whether or not to use KEEPFILTERS but, before you get to a rule like «I always insert it», please think about it as sometimes you do not want it!

CALCULATE Algorithm

We have already seen this….

```
1 Sales M Customers CALCULATE Intersect =
2 CALCULATE(
3     [Sales],
4     KEEPFILTERS(Customer[Gender] = "M")
5 )
```

Gender	Sales	Sales M Customers	Sales M Customers CALCULATE Intersect
F	4.932.322		
M	4.858.738	4.858.738	4.858.738
Totale	**9.791.060**	**4.858.738**	**4.858.738**

CalendarYear
- ☐ 2001
- ☐ 2002
- ■ 2003
- ☐ 2004

Filter Context in which [Sales], within CALCULATE, is executed

Filter Context in which [Sales M Customers CALCULATE] is executed

Gender	CalendarYear
F	2003

Gender
M

Here KEEPFILTERS is NOT what we want in the case in which we want to evaluate the ratio between the Sales to Customers of one Gender and all the Sales of the Customers of one Gender

CALCULATE Algorithm

We have already seen this….

```
1 Sales M Customers CALCULATE =
2 CALCULATE(
3     [Sales],
4     Customer[Gender] = "M"
5 )
```

Gender	Sales	Sales M Customers	Sales M Customers CALCULATE
F	4.932.322		4.858.738
M	4.858.738	4.858.738	4.858.738
Totale	**9.791.060**	**4.858.738**	**4.858.738**

CalendarYear
- 2001
- 2002
- 2003
- 2004

Filter Context in which [Sales], within CALCULATE, is executed

Filter Context in which [Sales M Customers CALCULATE] is executed

Ge	CalendarYear
F	2003

Gender
M

Now we have the desired result!

Contacts

francesco.bergamaschi@kubisco.com (business)

francesco.bergamaschi@unibo.it (students)

https://it.linkedin.com/in/francescobergamaschi

kubisco www.kubisco.com

Italy Power BI User Group

PBIG
POWER BI GEBRUIKERSGROEP

Session Feedback

Event Feedback