Función FILTER

Compatibilidad

Microsoft Excel
Excel ≥ 2010

★★★★★

Power BI Desktop
PBI ≥ Nov 2016

★★★★★

SQL Analysis Services
SSAS ≥ 2012

DIRECTQUERY: C.Calculadas ✓ Medidas ✓
ROW LEVEL SECURITY: ✓

Int. Contexto

Contexto de Filtro
Tiene en cuenta el contexto de filtro

★★★★★

Contexto de Fila
Ignora contextos de filas previos
Tiene en cuenta su contexto de fila

Categorías

Según Proceso Interno
Iteración

★★★★★

Según Resultado
Tabla

Recursos de Aprendizaje

EXPERTO EN LEGUAJE DAX
→ [Capacitación OnLine] ←
http://bit.ly/3P8GPCF

FILTER: Dragon de la Destrucción

Descripción

La función FILTER devuelve una tabla que representa un subconjunto de otra tabla o expresión de tipo tabla, es decir, retorna una tabla filtrada indicada en su primer parámetro de acuerdo con el citerior proporcionado en su segundo parámetro.

Sintaxis

1 Primer Parámetro — Segundo Parámetro 2

```
FLTER ( < Tabla > , < Expresión de Filtro > )
```

1 Tabla

Una tabla o expresión de tipo tabla a extraer un subconjunto de filas (filtrar).

ⓘ Iterador

Tipo
Obligatorio

Atributo
No Repetible

2 Expresión de Filtro

Una expresión booleana (TRUE/FALSE) que será evaluada para cada fila en la tabla.

ⓘ Contexto de Fila

Tipo
Obligatorio

Atributo
No Repetible

Valor Que Retorna

Una tabla completa o una tabla con una o más columnas, que contiene las filas filtradas.

OBSERVACIONES

I. La función FILTER puede filtrar filas en una tabla utilizando cualquier expresión valida booleana (TRUE/FALSE) en el contexto de fila.

Gracias a la transición de contexto utilizar una medida en el parámetro *Expresión de Filtro* es posible, con ello, filtramos la tabla basada en un cálculo dinámico involucrando otras filas y/o tablas.

Ejemplos

- Ejemplo 1 – Referencia Simple

Como señala su sintaxis en el primer argumento podemos llamar a cualquier tabla existente en el modelo, ejemplo:

```
1. PedidosNormal = -- Expresión para una Tabla Calculada
2. FILTER (
3.     Pedidos,
4.     Pedidos[Tipo Compra] = "Normal"
5. )
```

Incluso podemos llamar a una tabla calculada:

```
1. PedidosNormal = -- Expresión para una Tabla Calculada
2. FILTER (
3.     PedidosNormal, -- Tabla calculada previa
4.     Pedidos[País] = "Colombia"
5. )
```

- Ejemplo 2 – Expresiones de Tabla con Múltiples FILTER

Por otra parte, hay que notar que el primer argumento: ***admite una tabla***,

Lo anterior no implica que sea sólo una llamada simple de una tabla existente en el modelo, sino que también podemos indicar en dicho argumento cualquier función que retorne una tabla.

— *Ejemplo concreto:*

Si queremos la tabla únicamente para el *Tipo de Compra* igual a *Normal* y para que el *País* igual a *Colombia*, en lugar de crear primero una tabla calculada con un filtro inicial y luego llamarla para aplicar un segundo filtro *(como el ejemplo anterior)*, podríamos aplicar dos FILTER anidados *(aunque no es lo óptimo es un posible enfoque)* de la siguiente manera:

Vídeos Sobre FILTER

Función FILTER Hecho Fácil e Iteración

En el vídeo de YouTube **Miguel Caballero** muestra que entender la función FILTER en una primera instancia es sencillo.

Sin embargo, comprender su procedimiento interno *(mecanismo de iteración)* para retornar dicha tabla filtrada es aún mejor, ya que así tendremos la libertad de crear expresiones que devuelvan ciertas filas con expresiones booleanas más elaboradas.

https://bit.ly/3svRgDM

Condiciones Tipo O con FILTER

Cuando comprendemos que el **segundo argumento de la función FILTER** es una expresión de tipo booleano, empezamos a realizar condiciones con más elementos, en este vídeo se realizan algunos ejemplos.

https://bit.ly/3rqVlYv

```
1.  PedidosNormalyColombia =
2.  -- Expresión para una Tabla Calculada
3.  FILTER (
4.      FILTER (
5.          Pedidos,
6.          Pedidos[Tipo de Compra] = "Normal"
7.      ),
8.      Pedidos[País] = "Colombia"
9.  )
```

- Ejemplo 3 – Otras Funciones de Tipo Tabla

La función RELATEDTABLE permite devolver una tabla relacionada en el lado de los muchos, es decir, la implementamos en una tabla de dimensión.

Para verle un poco la aplicabilidad *(y de forma fugaz)*, podemos suponer que necesitamos en una tabla de dimensión donde tenemos la lista de productos *(SKUs)*, el número de unidades vendidas mayores a uno. En una en una nueva columnada calculada, una posible solución sería:

```
1.  #Vts>1xPrdct = -- Expresión para una Columna Calculada
2.  COUNTROWS (
3.      FILTER (
4.          RELATEDTABLE ( Pedidos ),
5.          Pedidos[Unidades] > 1
6.      )
7.  ) -- La tabla es de dimensión y contiene los SKUs
```

- Ejemplo 4 – Parámetro Expresión, Disyunción Lógica (O)

Imaginemos que necesitamos el número de ventas del producto "CB01" en conjunto con el producto "CC01".

Un camino común que tomarían los usuarios de *Excel* sería con la función del mismo nombre, OR, con lo cual la solución acompañada de COUNTOROWS sería así:

Más Vídeos Sobre FILTER

Uno de los conceptos más importante en el Lenguaje DAX son los tres pasos primordiales, ellos no ayudan a tener un entendimiento pleno del lenguaje para el análisis de datos.

En el vídeo de YouTube Miguel Caballero plantea una pregunta que incluirá la función FILTER y medidas, para así afinar nuestro conocimiento del lenguaje.

¿Acertaras?

https://youtu.be/cfbFpjm-6Xs

En el vídeo se estudian los fundamentos de la función FILTER utilizando Excel como lenguaje de consulta.

https://youtu.be/Btm6XX7TmFc

```
#VtsCB01oCC01 =
-- Expresión para una Medida
COUNTROWS (
    FILTER (
        Pedidos,
        OR (
            Pedidos[SKU] = "CB01",
            Pedidos[SKU] = "CC01"
        )
    )
)
```

- Ejemplo 5 – Disyunción Lógica: Múltiples "O"

Si necesitamos más de dos pruebas lógicas tipo O, no es viable con una sola función OR, dado que esta acepta sólo dos parámetros, con lo cual, una alternativa sería anidar dos funciones OR.

A pesar de lo previo, en el leguaje DAX contamos con el operador de disyunción lógica u operador O, el cual es representado con dos *plecas o barra vertical (pipe):* **||** , al ser un operador no tenemos límites para su implementación.

Ejemplo especifico: el número de ventas de los productos "CB0", "CC01", "L02" y "L07" en conjunto. Una vía de solución sería:

```
#VtsTop4 =
-- Expresión para una Medida
COUNTROWS (
    FILTER (
        Pedidos,
        Pedidos[SKU] = "CB01"
            || Pedidos[SKU] = "CC01"
            || Pedidos[SKU] = "L02"
            || Pedidos[SKU] = "L07"
    )
)
```

- Ejemplo 6 – Múltiples "O" con el operador IN

La medida anterior se puede escribir de forma más elegante y compacta gracias al operador IN, así:

```
#VtsTop4 = -- Expresión para una Medida
COUNTROWS (
        FILTER (
                Pedidos,
                Pedidos[SKU] IN { "CB01", "CC01", "L02" }
        )
)
```

- Ejemplo 7 – Aplicación de FILTER con Operador Tipo NOTIN

Imaginemos que ahora necesitamos el número de ventas de todos los productos, pero excluyendo "CB01" y "CC01".

El Operador NOTIN No Existe en el Lenguaje DAX.

Ahora bien, si el catálogo de productos consiste en 1000 SKUs, crear la medida señalando aquellos productos que si van incluido sería bastante tedioso.

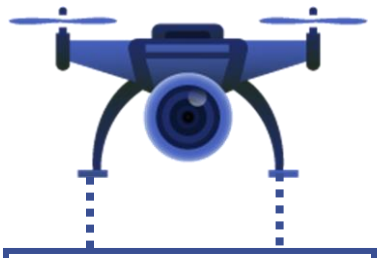
— *Tendríamos que listar 998 productos en el IN.*

Si embargo, no perdamos de vista que el segundo parámetro es una prueba lógica que devuelva TRUE o FALSE para indicar si dicha fila se incluye en resultado final.

Eso quiere decir, que, si encontramos la forma de invertir ese orden, es decir, si devuelve FALSE que lo convierta en TRUE, y si devuelve TRUE que lo convierta en FALSE, entonces, haríamos la tarea más fácil.

— *¿Qué otra función sino es NOT para abraza este resultado?*

Por lo tanto, aunque no existe el operador **NOTIN** su implementación se puede aplicar así:



Puedes estudiar el operador **IN** y cómo aplicar el operador **NOTIN** en el siguiente vídeo:

CLIC AQUÍ

```
#VtsNotIN = -- Expresión para una Medida
COUNTROWS (
    FILTER (
        Pedidos,
        NOT ( Pedidos[SKU]
            IN {
            "CB01",
            "CC01"
            } )
    )
)
```

- Ejemplo 8 – Conjunción Lógica: Múltiples "Y"

La necesidad del *"Y"* o AND no es extraño, por ejemplo, si queremos el número de ventas para *Tipo de Compra* igual *Normal* pero siempre y cuando el *País* sea igual a *Colombia*, en este caso la aplicación de *"Y"* es válido porque es de columna diferentes, esto quiere decir, que en la iteración actual se pueden cumplir o no los dos criterios.

La solución aplicando el operador Y que representa con el símbolo: **&&**, ya que AND al igual que OR sólo admite de dos parámetros, y, por lo tanto, es bueno conocer lo de una vez, la solución sería así:

```
#VtsNormalyColombia =
-- Expresión para una Medida
COUNTROWS (
    FILTER (
        Pedidos,
        Pedidos[Tipo de Compra] = "Normal"
            && Pedidos[País] = "Colombia"
    )
)
```

Para tu documentación la anterior expresión con la función AND sería de la siguiente manera:

```
#VtsNormalyColomiba =
-- Expresión para una Medida
COUNTROWS (
    FILTER (
        Pedidos,
        AND (
            Pedidos[Tipo de Compra] = "Normal",
            Pedidos[País] = "Colombia"
        )
    )
)
```

Se ha descrito que es posible hacer referencia a una tabla existente en el modelo en el primer parámetro, opción intuitivita y común. Sin embargo: sólo utilizar la referencia a una tabla existente en modelo si estrictamente necesario y no es viable otra solución, de lo contrario evitarlo a como dé lugar.

Caso Tabla Filtrada como Iterador

Si se desea determinar el promedio de los costos de producción sólo para aquellos SKU cuya primera letra sea la C, una alternativa no óptima puede ser:

```
AVERAGEX (
    FILTER ( Pedidos, LEFT ( Pedidos[SKU] ) = "C" ),
    Pedidos[Costo del Producto]
        + Pedidos[Costo Empaque]
)
```

No obstante, se esta utilizando FILTER para iterar en toda la tabla *Pedidos (referencia a tabla completa)*, donde, además, dicho resultado es utilizado por AVERAGEX que es otra función de iteración. Esto deriva en en iteraciones anidadas que al final del día desemboca en un plan de consultas no óptimo, en este caso y siempre que sea posible la tabla iterada por FILTER (*Pedidos* en este caso) debería ser filtrada mediante el contexto de filtro, moviendo las condiciones de filtro a los parámetros de filtros de CALCULATE. Una mejor solución es como sigue:

Acerca de las Cartas DAX

Las cartas DAX del equipo de **Power Skill** es un paquete de contenido de documentación y representación para un juego de todas las funciones en leguaje DAX, compuesta por dos partes:

I. La Carta

Cada función en todo el lenguaje DAX contará con un **personaje representativo**, por ejemplo, la función SUMX será representada por el ser mitológico: el grifo.

II. La Ficha Técnica

La ficha técnica tiene **información de la función** para su manejo, consulta y entendimiento, en ella se documenta y explica: Descripción, sintaxis, parámetros y más. (Cómo la presente)

Más Información:

→ https://bit.ly/3aZiBqu ←
→ www.CartasDax.Com ←

Última Actualización:
08 de septiembre del 2023

```
1.  PromedioDeCostosDeProduccionSoloC =
2.  CALCULATE (
3.      AVERAGEX (
4.          Pedidos,
5.          Pedidos[Costo del Producto] + Pedidos[Costo Empaque]
6.      ),
7.      LEFT ( Pedidos[SKU] ) = "C"
8.  )
9.
```

En esta situación se remueve incluso la necesidad de FILTER. Ahondar estos detalles requiere de capacitaciones o cursos formales como **Experto en Lenguaje DAX** para un entendimiento pleno o herramientas de optimización de código DAX como **DAX Optimizer** si se quiere ir al grano rápidamente

BIBLIOGRAFÍA

Páginas Web y Artículos:

- 1. DAX GUIDE: https://dax.guide/filter/
- 2. MICROSOFT: https://docs.microsoft.com/en-us/dax/filter-function
- 3. EFB: https://www.excelfreeblog.com/funcion-filter-en-dax-tratado-completo/
- 4. SQLBI: https://www.sqlbi.com/articles/filtering-tables/
- 5. JOSÉ POMARES: https://www.linkedin.com/pulse/la-funci%2525C3%2525B3n-filter-de-dax-y-su-peligroso-argumento

Libros:

- Definitive Guide To DAX (2nd Edition) – Marco Russo y Alberto Ferrari
- Practical PowerPivot & DAX Formulas – Art Tennick

Creado por:
Miguel Caballero y Fabian Torres.
Cualquier Retroalimentación:
excelfreebymcs@gmail.com

Funciones Relacionadas:

CALCULATETABLE