Función CALCULATE

Compatibilidad

Microsoft Excel
Excel ≥ 2010
★★★★★

Power BI Desktop
PBI ≥ Nov 2016
★★★★★

SQL Analysis Services
SSAS ≥ 2012

DIRECTQUERY: C.Calculadas ✗ Medidas ✓
ROW LEVEL SECURITY: ✓
CÁLCULOS VISUALES: ✓

Int. Contexto

Contexto de Filtro
Lo Tiene en Cuenta (al principio) **Luego**
Modifica el Contexto de Filtro
★★★★★

Contexto de Fila
Lo Tiene en Cuenta (al principio) **Luego**
Invalida Contextos de Fila

Categorías

Según Funcionamiento Interno
Indeterminado
★★★★★

Según Resultado
Escalar

Recursos de Aprendizaje

MAGÍSTER EN LEGUAJE DAX
Curso Pre-Grabado Completo:
→ [Curso Aquí] ←
bit.ly/3xB5WcD

Descripción

La función CALCULATE retorna la evaluación de una expresión escalar en un contexto de evaluación modificado *(bien sea contexto de filtro, contexto de fila o contexto visual)*, este contexto es transformado programablemente mediante filtros explícitos e implícitos, además, de los modificadores. ***¡Nota!*** *El contexto de evaluación, independientemente de su tipo, permite que una misma fórmula devuelva valores diferentes, ya que se ejecuta en subconjuntos distintos de filas.*

Sintaxis

1 Primer Parámetro — Segundo Parámetro (Opcional) 2

```
CALCULATE ( < Expresión > , [ <Filtro_1 > ] , [ <Filtro_2> ] , . . . )
```

3 Tercer Parámetro (Opcional)

1 Expresión

La expresión para evaluar en el contexto modificado.

ⓘ Expresión Escalar de Cualquier Tipo

Tipo
Obligatorio

Atributo
No Repetible

Observaciones

I. Aunque la expresión es el primer parámetro de la función CALCULATE, en realidad es el **último en ejecutarse**, dado que la modificación del contexto y/o el modelo lo llevan a cabo los parámetros del segundo en adelante.

En el siguiente ejemplo, la suma de la columna ingresos: `SUM ( Pedidos[Ingresos] )` es el segundo y último parámetro en calcularse, puesto que, el segundo parámetro: `ALL ( Pedidos )` es el primero.

```
1. =                -- Segundo en ejecutarse    -- Primero en ejecutarse
2. CALCULATE ( SUM ( Pedidos[Ingresos] ) , ALL ( Pedidos ) )
```

VIII. **Cualquier función o expresión escalar** es válida en el primer parámetro de la función CALCULATE, sin importar el tipo de dato que retorne, siempre y cuando devuelva un valor único.

- Ejemplo 1 - Expresión Sencilla:

```
1. #TodasVentas =
2. CALCULATE ( COUNTROWS ( Pedidos ) , ALL ( Pedidos ) )
```

- Ejemplo 2 - Expresión con más Elementos:

```
1. PromedioSemanasDeEntrega =
2. CALCULATE ( AVERAGEX ( Pedidos ,
3.                        DATEDIFF ( Pedidos[FechaEnvio] ,
4.                                   Pedidos[FechaLlegada], WEEK )
5.                        ) , Pedidos[SKU] = "CB01"
6.             )
```

Al ser CALCULATE una expresión que retorna un escalar se deduce que se pueden anidar múltiples CALCULATE.

- Ejemplo 3 - Expresión con Anidación de Varios CALCULATE:

```
1. IngColombia =
2. CALCULATE (
3.     CALCULATE ( [Ingresos Tot] , Pedidos[País] = "Colombia" ) ,
4.     Pedidos[País] = "Perú"
5. )
```

Se debe prestar especial atención al orden de ejecución de los CALCULATE, como se mencionó en la observación I: El primer parámetro es el último en ejecutarse, por lo tanto, la medida anterior retorna los ingresos para Colombia y no para Perú como se puede pensar en un primer momento.

Otros Recursos de Aprendizaje

El ADN de Power Pivot
Capítulo número 8
→ [Visitar Libro] ←

Int. Excel y Power BI
Capítulo número 17
→ [Visitar Libro] ←

DAX de 0 a Guerrero
Lección 6
→ [Visitar Curso] ←

NOTA IMPORTANTE:

INYECTAR FILTROS

A partir de aquí leerás el término: **Inyectar Filtros** al contexto de filtro, para simplificar, tómalo cómo: **Agregar Filtros o Restricciones a la Medida en CALCULATE**.

III. **El primer parámetro de la función CALCULATE es el único obligatorio.** Cuando se utiliza de esta manera se busca añadir *filtros implícitos*. Esta operación es denominada: *Transición de Contextos*, la cual se activa siempre que exista un contexto de fila generado en primer lugar.

- Ejemplo:

```
1. IngresosDelProducto = -- Expresión para columna calculada
2. CALCULATE ( SUM ( Pedidos[Ingresos] ) )
```

Un poco acerca de la transición de contextos.

Transición de Contextos

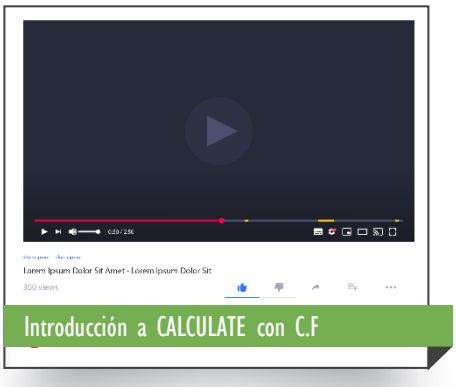
La transición de contextos consiste en inyectar filtros implícitos, moviendo un contexto de fila a un contexto de filtro, mediante dos operaciones: (1) invalidando cualquier contexto de fila existente, y luego, (2) agregando como argumentos de filtros implícitos (fantasma) todas las columnas y sus valores en la iteración actual. Por ello, debe generarse un contexto de fila implícito (automático) o explícito (programable) previamente.

Existen 6 consideraciones imprescindibles de la transición de contextos

i. Invalida cualquier contexto de fila.
ii. Agrega argumentos de filtro implícitos u ocultos (fantasma).
iii. No hay garantía de filtrar una fila individual.
iv. Es una operación de alto costo.
v. Los argumentos de filtros son de columnas nativas y calculadas.
vi. Creación de contexto de filtro a partir de contextos de fila.

Para aprender más: *Magíster en Lenguaje DAX*

Vídeos Sobre CALCULATE



Introducción a CALCULATE con C.F

En el vídeo de YouTube **Miguel Caballero** proporciona un vistazo de lo que es una **explicación más acertada de la función CALCULATE para el lenguaje DAX**, ya que, si bien existen muchos métodos de explicación los cuales buscan poder dar uso a CALCULATE de manera rápida, debemos ser 100% consientes que son explicaciones aproximadas.

→ http://bit.ly/2MKgzP5 ←

Iniciación a CALCULATE – Máster en DAX

El siguiente vídeo pertenece al curso **Máster en DAX y Power Pivot.**

→ http://bit.ly/31513Cj ←

No esta demás resaltar que el comportamiento de la transición de contexto se activa siempre que se genera un contexto de fila en primer lugar, y luego la expresión a ejecutar tenga un CALCULATE, sin importar si sólo se indicó el primer parámetro o tiene más de uno, la clave es que exista un CALCULATE.

CALCULATE IMPLICITO

Hay un comportamiento especial del lenguaje DAX que vale la pena resaltar siempre que sea posible: Cada vez que se hace referencia a una medida no importa si es desde otra medida o desde una columna calculada, la expresión interna queda encerrada en un CALCULATE.

```
1. Ingresos Tot =
2. -- Si primero se crea/define la siguiente medida:
3. SUM ( Pedidos[Ingresos] )
```

Luego es llamada en algún cálculo DAX, ejemplo:

```
1. IngresosDelProducto = [Ingresos Tot]
2. -- Sólo se llama a la medida, aquí desde una columna calculada
```

Entonces la expresión interna queda:

```
1. IngresosDelProducto = -- Al expandir la medida referenciada
2. CALCULATE ( SUM ( Pedidos[Ingresos] ) )
```

Por este CALCULATE implícito es tan importante la convención para distinguir siempre cuando se hace referencia a una medida y a una columna de una tabla, indicando una medida sin la tabla donde reside, así: [Ingresos Tot] y una columna siempre indicando la columna donde reside, así: Pedidos[Ingresos], **para ser consientes en todo momento de una medida, de tal manera que si se genera un contexto de fila previo podamos notar la transición de contextos.**

Artículo de CALCULATE

En el artículo se describe la función CALCULATE, hay que recalcar, que esta explicación simplemente es una aproximación para conocer la función CALCULATE, **no es en ninguna circunstancia la explicación precisa y exacta**, su entendimiento requiere de teoría, varios pasos esenciales y ejercicios para comprender, honestamente, como opera la función CALCULATE, esto se puede conseguir con el **Énfasis en CALCULATE** o el vídeo curso en el **Máster en DAX y Power Pivot**.

→ http://bit.ly/318mToD ←

Parámetros o Argumentos de Filtro

Del parámetro dos en adelante se repite, y en su conjunto son llamados: argumentos o parámetros de filtro.

1 Filtro_1

Tipo
Opcional

Atributo
Repetible

Este parámetro tiene dos facetas:

FACETA I: COMO FILTRO

Expresión de tabla que representa un filtro, el cual se inyecta como nueva restricción para el contexto. Comúnmente, esta expresión de tabla se indica como una expresión booleana (TRUE/FALSE), no obstante, no es más que un azúcar sintáctico (syntax sugar) para hacer las expresiones de tabla más fáciles de leer.

FACETA II: COMO MODIFICADOR

Mediante la ayuda de unas funciones especiales llamadas modificadores, el parámetro ya no actúa como filtro para ser inyectado al contexto, sino que cambia su comportamiento para manipular el modelo de datos, así:

- Cambios en la arquitectura de las relaciones mediante: USERELATIONSHIP y/o CROSSFILTER.

- Cambios en la estructura original del contexto mediante: (1) ALL, ALLCROSSFILTERED, ALLEXCEPT, ALLNOBLANKROW y REMOVEFILTERS para remover filtros del contexto de filtro, (2) ALLSELECTED para restaurar filtros en el contexto de filtro, y (3) KEEPFILTERS para mantener filtros nativos del contexto de filtro *(aunque este último es propiamente un modificador de parámetro - Véase las observaciones-)*

Modificadores a Nivel de Modelo

- Cambios en el nivel del entramado de la tabla virtual: EXPAND y EXPANDALL para subir de nivel, y COLLAPSE y COLLAPSEALL para bajar de nivel.

Modificadores a Nivel de Visual

Observaciones

FACETA I: COMO FILTRO

Inyectar un Filtro al Contexto

I. Si se desea **inyectar un filtro adicional al contexto**, a menudo se expresa como expresión booleana (TRUE/FALSE), por ejemplo, si desea agregar un filtro para el SKU con nombre CB01, se haría así:

```
IngCB01 =
CALCULATE ( [Ingresos Tot] , Pedidos[SKU] = "CB01" )
```

No obstante, no se debe olvidar que la expresión: Pedidos[SKU] = "CB01" es solo una notación de tipo azúcar sintáctico para hacer la expresión más dulce, por esto, dicha expresión se convierte realmente en: FILTER (ALL (Pedidos[SKU]) , Pedidos[SKU] = "CB01") con lo cual la medida IngCB01 es en realidad:

```
IngCB01 =
CALCULATE ( [Ingresos Tot] ,
            FILTER ( ALL ( Pedidos[SKU] ) ,
                     Pedidos[SKU] = "CB01" )
)
```

La syntax sugar (azúcar sintáctico) no tiene ninguna diferencia entre rendimiento y procesamiento interno, es exactamente lo mismo, sin embargo, es recomendable utilizar la syntax sugar siempre que sea posible sin perder de vista en ningún momento la expresión de tabla (FILTER/ALL) detrás que la compone.

Nota Sobre Syntax Sugar

El orden en el azúcar sintáctico es permutable, con lo cual Pedidos[SKU]="CB01" es igual que "CB01"= Pedidos[SKU].

Inyectar Varios Filtros (Actúan como un Y)

II. Si se desea **inyectar varios filtros al contexto que actúen como una conjunción lógica**, dicho de otro modo, como un **"y"**, cada filtro se debe indicar en su propio parámetro de filtro.

Cuando corresponde a elementos de columnas diferentes, su uso es intuitivo:

- Ejemplo:

```
IngCB01yColombia =
CALCULATE ( [Ingresos Tot] ,
            Pedidos[SKU] = "CB01" ,
            Pedidos[País] = "Colombia"
)
```

A razón de lo anterior, se puede deducir que filtros que afecten a la misma columna con un solo elemento, y que, se han implementado en diferentes parámetros de CALCULATE, siempre retornaran vacío.

- Ejemplo:

```
IngVacío =
CALCULATE ( [Ingresos Tot] ,
            Pedidos[SKU] = "CB01" ,
            Pedidos[SKU] = "CC01"
)
```

No se debe asumir siempre que varios argumentos de filtros que afecten a la misma columna van a retornar vacío, debido a que **la operación que se realiza entre ellos es la intersección de sus elementos antes de la aplicación al modelo**, por lo tanto, si comparten elementos si devuelven algún valor.

- Ejemplo:

```
IngCB01-Intersección =
CALCULATE ( [Ingresos Tot] ,
            Pedidos[SKU] IN {"CB01" , "CC01"} ,
            Pedidos[SKU] IN {"L07" , "CB01"} )
-- la intersección de los elementos de los dos filtros es CB01
```

La medida **IngCB01** es orientada a ilustrar académicamente el comportamiento de la intersección de una manera más visual. Una medida con mayor probabilidad de aplicación con la ayuda de la intersección puede ser como sigue:

NOTA

SOBRE DESIGUALDADES

La desigualdad de la medida IngEntre30y50 pude ser escrita en un único parámetro de filtro, una primera posibilidad:

```
Pedidos[Ingresos] >= 30 &&
Pedidos[Ingresos] <= 50
```

Una segunda posibilidad:

```
AND ( Pedidos[Ingresos] >= 30 ,
      Pedidos[Ingresos] <= 50 )
```

- Ejemplo:

```
IngEntre30y50 =
CALCULATE ( [Ingresos Tot] ,
            Pedidos[Ingresos] >= 30 ,
            Pedidos[Ingresos] <= 50
) -- Especialmente útil para desigualdades
```

La descripción hasta aquí corresponde a cómo interactúan filtros generados por CALCULATE, lo que nos lleva a cómo interactuar los filtros creados por CALCULATE (programables) con los filtros nativos (automáticos).

Se debe tener presente que, si en el contexto de filtro original hay un filtro que afecte a la misma columna por uno inyectado por los parámetros de filtros de CALCULATE, entonces, el de CALCULATE se impone sobre el nativo sobrescribiéndolo. La manera de evitar este comportamiento y hacer que los dos convivan, es utilizando un modificador de parámetro de CALCULATE, particularmente: KEEPFILTERS.

- Ejemplo:

```
IngCB01 =
CALCULATE ( [Ingresos Tot] ,
            Pedidos[SKU] = "CB01" ,
            KEEPFILTERS ( Pedidos[País] = "Colombia" )
)
```

Inyectar un Filtro con Varios Elementos (Actúan como un O)

III. Si se desea **inyectar un filtro al contexto que actúen como una disyunción lógica**, es decir, como un **"O"**, se debe indicar en un sólo parámetro donde el filtro tenga en sus elementos los valores que constituyen el O.

- Por ejemplo: Una medida que devuelva los ingresos para los productos: CB01, CC01 y L02; puede ser escrita:

```
IngCB01oCC01oL02 =
CALCULATE( [Ingresos Tot], Pedidos[SKU] IN {"CB01", "CC01", "L02" })
```

La expresión de la medida también puede ser escrita:

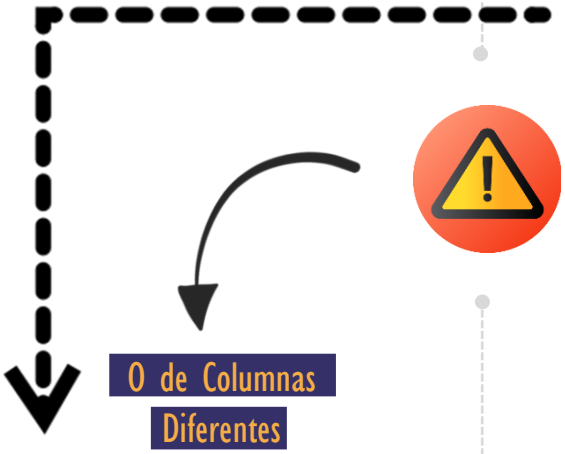
```
1.  IngCB01oCC01oL02 =
2.  CALCULATE (
3.      [Ingresos Tot] ,
4.      Pedidos[SKU] = "CB01"
5.          || Pedidos[SKU] = "CC01"
6.          || Pedidos[SKU] = "L02"
7.  ) -- incluso puede ser escrita con dos OR anidados.
```

Aplicar un O de elementos de columnas diferentes es posible con azúcar sintáctico, pero **sólo para versiones de Power BI**, SSAS y Azure Analysis Posteriores a febrero del 2021.

Por lo tanto, la siguiente expresión:

```
1.  IngCB01oColombia =
2.  CALCULATE ( [Ingresos Tot] ,
3.                  Pedidos[SKU] = "CB01"
4.                      || Pedidos[País] = "Colombia"
5.  )
```

RETORNA ERROR PARA POWER BI ANTES DE MARZO DEL 2021 Y POWER PIVOT.



O de Columnas Diferentes

La expresión O de columnas diferentes en azúcar sintáctico para tecnologías válidas: *(Power, SSAS y AZ posterior febrero 2021)* se convierte internamente en la versión de la primera medida presentada en la página siguiente.

Opción No. 1: PELIGORSA

En este caso, la expresión de tipo tabla debe ser redactada de forma explícita, una posibilidad puede ser:

```
1.  IngCB01oColombia =
2.  CALCULATE ( [Ingresos Tot] ,
3.                  FILTER ( Pedidos , Pedidos[SKU] = "CB01"
4.                                  || Pedidos[País] = "Colombia" )
5.  ) -- Esta expresión es válida: pero NO RECOMENDABLE.
```

Empero, ES UNA EXPRESIÓN PELIGROSA, ya que es un filtro de tabla y no de columna, alto impacto en el rendimiento y puede arrojar valores imprecisos, en definitiva: una muy mala práctica…

…Una opción segura y mucho más apropiada sería como sigue:

Opción No. 2: CARDINALDAD DE COLUMNA ALTA

```
1.  IngCB01oColombia =
2.  CALCULATE (
3.      [Ingresos Tot] ,
4.      FILTER (
5.          ALL ( Pedidos[SKU] , Pedidos[País] ) ,
6.          Pedidos[SKU] = "CB01"
7.           || Pedidos[País] = "Colombia"
8.      )
9.  ) -- Recomendable para cardinalidad de columnas alta.
```

Se ha utilizado ALL con dos columnas para crear la lista de combinaciones existentes de las columnas proporcionadas en sus parámetros. [VÉASE LA FICHA TÉCNICA DE ALL]. La cardinalidad del filtro puede ser más pequeña que el producto cartesiano de los elementos de las columnas.

Si la cardinalidad de las columnas es pequeña, la expresión anterior puede ser de un rendimiento bajo, puesto que ALL debe escanear toda la tabla buscando las combinaciones; para dicho caso, una alternativa es la siguiente:

Opción No. 3: CARDINALDAD DE COLUMNA BAJA

```
1.  IngCB01oColombia =
2.  CALCULATE (
3.      [Ingresos Tot] ,
4.      FILTER (
5.          CROSSJOIN ( ALL ( Pedidos[SKU] ), ALL ( Pedidos[País] ) ) ,
6.          Pedidos[SKU] = "CB01"
7.              || Pedidos[País] = "Colombia"
8.      )
9.  ) -- Recomendable para cardinalidad de columnas baja.
```

Con la función CROSSJOIN se obtiene todas las posibles combinaciones sobre todos los valores de las columnas, independiente si las combinaciones existen o no en el registro; en este caso, la cardinalidad del filtro es igual al producto cartesiano de los elementos de las columnas. [VÉASE LA FICHA TÉCNICA DE LA FUNCIÓN CROSSJOIN].

Inyectar un Filtro con Varios Elementos (Actúan como un O) De Columnas Diferentes en Tablas Diferentes

Si la aplicación del "O" se desea de columnas en tablas diferentes, la opción de la función ALL ya no es posible, puesto que únicamente admite columnas de la misma tabla, para esta situación se debe recurrir a la función SUMMARIZE.

La función SUMMARIZE genera la lista de combinaciones existentes entre dos o más columnas, y se pueden utilizar columnas de diferentes tablas siempre y cuando estén relacionadas de muchos a uno y que se puedan acceder por cascada (copo de nieve).

- Ejemplo:

```
IngCBOIoColSMZ =
CALCULATE (
    [Ingresos Tot] ,
    FILTER (
        SUMMARIZE ( Pedidos , Pedidos[Pais] ,
                                SKUProductos[NombreProducto] ) ,
        Pedidos[Pais] = "Colombia"
    || SKUProductos[NombreProducto] = "Batman Begins"
    )
)
-- Cardinalidad menor al producto cartesiano, pero se escanea la tabla
-- completa en busca de las combinaciones, lo cual puede llegar a ser
-- contraproducente para el rendimiento cuando las columnas tienen
-- baja cardinalidad y la tabla es de un número de registros alto.
```

Inyectar un Filtro con Varios Elementos (Actúan como un O) De Columnas Diferentes en Tablas Diferentes No Relacionadas

Si la aplicación del "O" se desea de columnas en tablas diferentes pero que no están relacionadas, la opción con la función SUMMARIZE no es factible, pues demanda que las tablas estén relacionadas.

Para esta situación se debe optar por la función CROSSJOIN en combinación con la función VALUES, aunque la alternativa con CROSSJOIN es más general, pues aplica estén o no relacionadas las tablas, es recomendable hacer pruebas de rendimiento cuando se pueda ejecutar con las dos opciones.

DAX.do

El servicio online DAX.do es un playground el cuál permite escribir y ejecutar expresiones de consulta DAX, para con ello hacer pruebas y experimentar sin la necesidad de crear un proyecto de Power BI o similar, o tener la necesidad de buscar algunas tablas, pues DAX.do ofrece todo ello para "jugar" y experimentar con las expresiones.

Eso sí, en DAX.do utilizamos el lenguaje DAX como lenguaje de consulta.

→ DAX.do ←

- Ejemplo:

```
1.  IngCBDoCoICORSSJOIN =
2.  CALCULATE (
3.      [Ingresos Tot],
4.      FILTER (
5.          CROSSJOIN (
6.              VALUES ( Pedidos[Pais] ),
7.              VALUES ( SKUProductos[NombreProducto] )
8.          ),
9.          Pedidos[Pais] = "Colombia"
10.             || SKUProductos[NombreProducto] = "Batman Begins"
11.     )
12. )
13.
14. -- Si las tablas están relacionadas y se opta por esta vía,
15. -- Siendo que las combinaciones existentes es menor al producto
16. -- Cartesiano de las columnas involucradas en la función CROSSJOIN,
17. -- Entonces, esta solución es contraproducente desde la perspectiva del
18. -- Rendimiento, no obstante, si no están relacionadas o no se pueden
19  -- Relacionar, entonces, esta solución será la alternativa más directa.
```

Cualquier expresión de tipo tabla (tabular) es legítima en los parámetros de filtros de CALCULATE y no se limita a las funciones previamente señaladas, los ejemplos clásicos son las funciones de inteligencia de tiempo, como: DATESYTD, DATESBETWEEN, DATEADD, LASTNONBLANK, etc., Etc.

Pero también funciones de tipo tabla como: ADDCOLUMNS, EXCEPT, INTERSECT, ROW, SELECTCOLUMNS, UNION, VALUES, WINDOW, etc., Etc.

E incluso crear los filtros con los constructores de tabla es viable apoyado de la función TREATAS para actualizar el data lineage.

De hecho, también se puede trabajar con filtros arbitrarios.

Para aprender más: *Magíster en Lenguaje DAX*

FACETA II: CÓMO MODIFICADOR

Modificadores de Arquitectura de las Relaciones

El primer conjunto de modificadores tiene como objetivo: **Alterar la arquitectura de las relaciones**, bien sea activando una relación o cambiando el filtro cruzado, esta categoría está compuesta por las funciones USERELATIONSHIP y CROSSFILTER.

I. Si se desea activar una relación para el tiempo de vida de ejecución de una expresión DAX, el modificador USERELATONSHIP permite llevar acabo esta tarea.

- Ejemplo:

```
Ingresos Llegada =
CALCULATE (
    [Ingresos Tot] ,
    USERELATIONSHIP ( Pedidos[FechaLlegada] , Calendario[Fecha] )
) -- En la función USERELATIONSHIP primero se indica la clave externa
  -- Y en el segundo parámetro la clave primaria, empero, también
  -- Se puede indicar viceversa.
```

No olvidar que la relación física debe existir previamente en la interfaz, puesto que USERELATIONSHIP la activa, si no está creada arroja error.

II. Si se desea alterar el filtro cruzado, esto se puede conseguir con el modificador CROSSFILTER, quien acepta cinco opciones en su tercer parámetro: BOTH o 2 (Bidireccional), ONEWAY o 1 (Unidireccional), NONE o 0 (desactiva la relación), ONEWAY_RIGHTFILTERSLEFT o 3 y ONEWAY_LEFTFILTERSRIGHT o 4.

La opción más utilizada es con BOTH, la cual es la opción recomendada si se necesita la propagación en ambos sentidos, en lugar de activar la relación bidireccional físicamente en la interfaz.

- Ejemplo:

```
PrDtsVend =
CALCULATE (
    COUNTROWS ( SKUProductos ) ,
    CROSSFILTER ( Pedidos[SKU] , Productos[SKU] , BOTH )
) -- Opción preferible siempre que sea posible.
```

Modificadores de Estructura del Contexto de Filtro

El segundo conjunto de modificadores tiene como objetivo: **Cambiar la estructura del contexto de filtro**, bien sea (1) removiendo filtros del contexto o (2) restaurando filtros al contexto.

(1) En la primera subcategoría (modificadores que remueven filtros) tenemos las funciones: ALL, ALLEXCEPT, ALLNOBLANKROW, ALLCROSSFILTERED y REMOVEFILTERS, esta última es sólo un alias de ALL para su segunda cara.

(2) En la segunda subcategoría (modificadores que restauran filtros) tenemos la función ALLSELECTED.

Es imperativo no dejarnos confundir por el comportamiento dual/silencioso de las funciones ALLxxxx, ya que tienen dos caras, la primera cuando se utiliza en cualquier expresión DAX que no la involucra de manera libre o cómo función de primer nivel en los parámetros de CALCULATE y CALCULATETABLE, aquí materializa la tabla. La segunda cara cuando se utiliza **cómo función de primer nivel en parámetros de filtros de CALCULATE** y CALCULATETABLE, es de esta manera cuando pasan a actuar como modificadores. [Véase la Ficha Técnica de ALL]

El ejemplo por excelencia de ALL cómo modificador para CALCULATE consiste en determinar el porcentaje de participación, el cual se compone del cálculo aritmético:

Valor Actual (Numerador) / Valor Total (Denominador)

- Ejemplo:

```
%Particiacion =
VAR Numerador =
    SUM ( Pedidos[Ingresos] )
VAR Denominador =
    CALCULATE ( SUM ( Pedidos[Ingresos] ) , ALL ( Pedidos ) )
VAR PctParticipacion =
    DIVIDE ( Numerador , Denominador )
RETURN
    PctParticipacion
```

La variable Denominador utiliza como parámetro de filtro la expresión: ALL(Pedidos), la cual remueve todos los filtros que afecten a la tabla Pedidos, en este caso, no se crea ninguna tabla, sino que manipula el contexto de filtro borrando listas de valores existentes.

Si lo que se busca es la participación respecto a un subtotal, bien sea a nivel de columnas o filas, se puede conseguir con un parámetro modificador y un parámetro de filtro, así:

```
%PartVisible =
DIVIDE (
    [Ingresos Tot] ,
    CALCULATE ( [Ingresos Tot], ALLSELECTED(), VALUES ( Pedidos[País] ) )
)
```

Si lo que se desea es tener en cuenta los filtros generados por objetos visuales por fuera de la visualización actual, entonces, es necesario la restauración de filtros, para esta tarea se precisa la función ALLSELECTED.

- Ejemplo:

```
%PartVisible =
DIVIDE ([Ingresos Tot], CALCULATE ([Ingresos Tot] , ALLSELECTED () ) )
```

NOTA

El entendimiento pleno y real de la función ALLSELECTED requiere del concepto de: *contexto de filtro sombra (shadow filter context)*, de hecho, le agrega más complejidad al lenguaje, a pesar de ello, la descripción aproximada presentada, la cual dice:

La función ALLSELECTED restaura o tiene en cuenta los filtros generados por los objetos visuales por fuera de la visualización actual.

Es bastante útil para su implementación en la gran mayoría de escenarios.

Vídeos Sobre Cálculos Visuales

Los **Cálculos Visuales en Lenguaje DAX** introducen una dimensión innovadora al interactuar con los datos directamente desde una interfaz visual, en lugar de operar a través del modelo subyacente. **Esta evolución representa un hito en DAX, al ser el primer enfoque de su tipo que enriquece el lenguaje con una gama ampliada de funciones y conceptos.** Similar a la importancia de comprender el contexto de fila, el contexto de filtro y la transición de contexto en los cálculos tradicionales, dominar la orientación del análisis —incluyendo el manejo del eje y la capacidad de reinicio— es fundamental en el ámbito de los Cálculos Visuales.

→ https://bit.ly/3zFjlRS ←

Este vídeo revela el parámetro **RESET** en cálculos visuales.

→ https://bit.ly/4eCrTI5 ←

Modificadores de Nivel de Entramado

El tercer conjunto de modificadores es para Cálculos Visuales, quienes tienen como como objetivo: **Cambiar el nivel en el entramado de la tabla virtual**, por un lado los que suben de nivel brindado más detalle *(mayor granularidad)*: EXPAND y EXPANDALL que a su vez suelen demandar un agregación, y por otro lado: COLLAPSE y COLLAPSEALL que suele ir a un detalle inferior.

- Ejemplo:

En una matriz donde el campo *País* está en el área de filas, y el campo *Tipo de Compra* (con los valores *"Normal"* y *"Devolución"*) también está en el área de filas, se desea mostrar el valor de las transacciones normales a nivel de *País*. Una posible solución sería:

```
#VentasNormales =
IF (
    NOT ISATLEVEL ( [Tipo de Compra] ),
    CALCULATE (
        SUMX (
      -- El uso de ROWS sin más ignora los filtros en el contexto visual.
        -- Por ello, es necesario utilizar VALUES ( ROWS )  para que se
        -- Conserve el filtro de País. Este comportamiento es opuesto
        -- Al de las medidas, por lo que es necesario en el entorno
        -- Visual.
          FILTER ( VALUES ( ROWS ), [Tipo de Compra] = "Normal" ),
          [#Ventas]
        ),
        EXPAND ( ROWS, 1 ) -- Sube 1 nivel en el entramado.
    ) -- No se utiliza el filtro de Tipo de Compra en un parámetro
)   -- De CALCULATE, porque dado el orden de precedencia al ser
    -- Construido se encuentra en el nivel del entramado original,
    -- Por lo tanto, la columna Tipo de Compra no estaría en el
    -- Entramado, creando un filtro cuyo data lineage no se
    -- Corresponde con nadie, aplicándose a una tabla anónima.
```

- Ejemplo:

En una matriz donde el campo *País* está en el área de filas, determinar el porcentaje de participación para *#Ventas*, la siguiente expresión DAX para cálculo visual brinda una solución:

```
%Particiacion =
DIVIDE (
    [#Ventas],
    CALCULATE (
        [#Ventas],
        COLLAPSE ( ROWS, 1 )
    )
)
-- Baja 1 nivel en el entramado.
```

Para el escenario particular planteado, también se obtiene el resultado de esta manera:

```
%Particiacion =
DIVIDE (
    [#Ventas],
    CALCULATE (
        [#Ventas],
        COLLAPSEALL ( ROWS )
    )
)
-- Baja al nivel más inferior en el entramado.
```

Por supuesto, las expresiones anteriores también se logran con la función ALL. Sin embargo, son ejemplos para mostrar brevemente COLLAPSE y COLLAPSEALL.

Para aprender más:  *Magíster en Lenguaje DAX*

Valor Que Retorna

Retorna el resultado de una expresión escalar de cualquier tipo evaluado en un contexto de evaluación modificado.

OBSERVACIONES

I. Hacer referencia a una medida utilizando syntax sugar en un parámetro de filtro no es posible, por lo tanto, la siguiente expresión no es válida:

```
1.  IngresosDeAltaRentabilidadxProducto =
2.  CALCULATE ( [Ingresos Tot] ,
3.                  Pedidos[Ingresos] > [Ing] * 0,8 ) -- Error!
```

Una manera de escribir esta medida puede ser así:

```
1.  IngresosDeAltaRentabilidadxProducto =
2.  CALCULATE (
3.    SUM ( Pedidos[Ingresos] ),
4.    FILTER ( ALL ( Pedidos[Ingresos] ), Pedidos[Ingresos] > [Ing] * 0,8 )
5.  )
```

II. Utilizar el modificador de parámetros KEEPFILTERS con una expresión explícita es 100% lícito, especialmente útil para condiciones "O", ejemplo:

```
1.  IngCB01oColombia =
2.  CALCULATE (
3.      [Ingresos Tot],
4.      KEEPFILTERS (
5.          FILTER (
6.              ALL ( Pedidos[SKU], Pedidos[País] ),
7.              Pedidos[SKU] = "CB01"
8.                  || Pedidos[País] = "Colombia"
9.          )
10.     )
11. )
```

NOTA

Una expresión explícita en los parámetros de filtros de CALCULATE se refiere a escribir la expresión tabular completa, ejemplo:

```
FILTER ( ALL(Pedidos[País]);
        Pedidos[País] = "Perú")
```

Y no en su forma resumida en *syntax sugar*, ejemplo:

```
Pedidos[País] = "Perú"
```

III. Los filtros creados programablemente por los parámetros de filtros de la función CALCULATE (inyectados) sobrescriben los filtros que afecten a la misma columna, que existan automáticamente generados por el reporte.

   a. Pueden existir expresiones con CALCULATE anidados, en este caso, los filtros del CALCULATE más interno sobrescriben a los filtros del CALCULATE más externo si afectan a la misma columna.

   b. La sobreescritura también ocurre si el filtro viene en una tabla acompañada de otras columnas (Sin importar si dicha columna se encuentra en el espectro expandido), puesto que CALCULATE siempre se impone sobre los filtros nativos, a menos que se indique lo contrario.

   c. Se deduce que pueden existir más de dos contextos de filtro a la vez, a parte del contexto de filtro original *(conocido también como contexto de filtro automático, contexto de filtro implícito o contexto de consulta)* y el contexto de filtro copia *(conocido también como contexto de filtro programable, contexto de filtro explícito, o simplemente contexto de filtro)*, ya que, si hay varios CALCULATE anidados, fácilmente puede existir 5, 6, 7 o más contextos de filtros.

IV. Si no se desea que un filtro programable sobrescriba a uno nativo, este se debe encerrar en la función de modificación de parámetros: KEEPFILTERS, lo mismo aplica si se desea que un filtro de un CALCULATE más interno no sobrescriba a un filtro de un CALCULATE más externo.

V. La función KEEPFILTERS se puede emplear indistintamente en diversos parámetros de filtros, por consiguiente, si se desea evitar la sobreescritura de un parámetro específico se implementa KEEPFILTERS, mientras que en otro parámetro de filtro se puede omitir.

VI. Los filtros que afectan a la misma columna y conviven gracias a KEEPFILTERS, primero crean la intersección de sus elementos antes de ser aplicados al modelo de datos.

No se debe confundir esta parte con la aplicación de filtros de columnas diferentes, Que en conjunto se aplican como un Y.

NOTA

El termino *Query Context* o *Contexto de consulta* es presentado en la documentación de Microsoft, terminología que *Miguel Caballero* de *Power Elite* también ha empleado, situación que se ve especialmente enmarcada en el libro:

EL ADN DE POWER PIVOT
http://eladndepowerpivot.com/

Sin embargo, en la actualidad raramente utilizamos este término, ya que tan sólo lo denominamos contexto de filtro original o automático, y los posteriormente creados, contextos de filtro copia o programable.

El término *restricción* o *constrain* utilizado ampliamente en el libro El ADN de Power Pivot utilizado para referirnos a los filtros generados por los parámetros de filtros de CALCULATE en la actualidad lo seguimos empleando.

MECANISMOS

Un MECANISMO es todo aquello que pude producir un filtro para ser inyectado al contexto de filtro. En consecuencia, tenemos los siguientes mecanismos:

- **Elementos de la Interfaz**: objetos visuales y panel de filtros. *(Todo parte de un mismo mecanismo).*
- Parámetros de filtros explícitos de CALCULATE.
- Parámetros de filtros explícitos de CALCULATETABLE.
- ~~Operación de generación de nuevas columnas SUMMARIZE.~~

  ⚠ NO RECOMEDADO, CONSIDERESE OBSOLETO!
- Parámetros de filtros de SUMMARIZECOLUMNS.
- La transición de contextos al convertir un contexto de fila en contexto de filtro que deriva en filtros implícitos.

ACLARACIÓN: Los filtros generados por un mismo CALCULATE son considerados de un mismo mecanismo, por lo tanto, filtros en CALCULATE(s) distintos se consideran diferentes mecanismos.

Los filtros implícitos se consideran parte de otro mecanismo, aunque se listen de forma oculta en el mimo conjunto de parámetros de filtro para un mismo CALCULATE, dado que surgen del contexto de fila (otro ente).

SOBREESCRITURA DE FILTROS

DOS FILTROS QUE AFECTAN A LA MISMA COLUMNA GENERADOS POR MECANISMOS DIFERENTES, ENTONCES, EL ÚLTIMO EN INGRESAS AL CONTEXTO SOBREESCRIBE.

INTERSECCIÓN DE FILTROS

DOS FILTROS QUE AFECTAN A LA MISMA COLUMNA GENERADOS POR MISMO MECANISMO OCURRE LA INTERSECCIÓN.

VII. Si dos o más filtros afectan a la misma columna, pero son generados por un mismo mecanismo, entonces, la intersección de sus elementos ocurre sin la necesidad de KEEPFILTERS, por ejemplo, al ser creados por distintos parámetros de filtros en un mismo CALCULATE, como es el caso de:

```
1. IngCB01 =
2. CALCULATE ( [Ingresos Tot] ,
3.             Pedidos[SKU] IN {"CB01", "CC01"},
4.             Pedidos[SKU] IN {"L07", "CB01"}" )
5. -- la intersección de los elementos de los dos filtros es CB01.
```

Nótese que los dos filtros afectan a la misma columna y son generados por un mismo CALCULATE (mismo mecanismo), es por ello por lo que se realiza la intersección de sus elementos antes de ser aplicado al modelo de datos, situación diferente si el filtro existe previamente por el reporte (contexto de filtro original) y luego llega uno de CALCULATE (contexto de filtro copia) que afecta a la misma columna **(generados por mecanismos diferentes)**.

a. Este escenario también se puede dar por filtros creados por el reporte (contexto de filtro original), por ejemplo, utilizando dos segmentaciones de datos de la misma columna. Un evento poco frecuente pero posible, a menos que sea con interacciones visuales.

   Sin embargo, algo más frecuente es que sea un filtro creado por una segmentación de datos y otro por un área de colocación.

VIII. La diferencia entre un contexto de filtro vacío y un filtro vacío, es bueno apuntarlo aquí, ya que se da con CALCULATE:

a. Un contexto de filtro vacío significa que las tablas del modelo de datos quedan intactas y todos sus registros son visibles, por lo tanto, una medida devolvería el cálculo general.

b. Un filtro vacío indica que para esa columna no hay ningún elemento visible, por lo que la tabla en cuestión y todas las relacionadas mediante el tipo 1 → * quedan completamente vacías, por lo tanto, una medida de alguna de esas tablas devolvería BLANK.

IX. Es posible utilizar los parámetros de filtros del CALCULATE implícito que surge de forma oculta cuando se llama a una medida.

Ejemplo: Crear una columna calculada en la tabla de dimensión *productos* que devuelva los ingresos por producto para el país *Colombia* (Columna que sólo existe en la tabla de hechos *Ventas*), teniendo en cuenta que:

- Productos = SKUProductos
- Ventas = Pedidos

La solución se consigue así:

```
=
CALCULATE (
        [Ingresos Tot] ,
        Pedidos[País] = "Colombia"
) -- El filtro de SKUProductos se crea implícitamente por
  -- La transición de contextos.
```

Pero puede ser rescrita con la *sintaxis acortada* del CALCULATE implícito en la medida [Ingresos Tot], así:

```
=
[Ingresos Tot] ( Pedidos[País] = "Colombia" )
```

Incluso se pueden implementar varios filtros, así:

```
=
[Ingresos Tot] (
    Pedidos[País] = "Colombia" ,
    Pedidos[Tipo de Compra] = "Normal"
)
```

No obstante, **LA SINTAXIS ACORTDA DEL CALCULATE IMPLÍCITO**, debe evitarse a toda costa, puesto que: El Demonio Esta en los Detalles, y DAX está lleno de detalles, por ello es mejor mantener a la vista CALCULATE…

CALCULATE tiene un montón de operaciones involucradas, y su orden de ejecución no es el intuitivo siguiendo del primero al último parámetro, en realidad sigue un orden establecido bien definido y exacto:

ORDEN DE PRECEDENCIA

i. Todos los parámetros de filtros explícitos son preparados individualmente en el contexto de evaluación original, es decir, contexto de filtro original y contextos de fila si existen (o contexto visual si es a nivel visual).

> Aquí los filtros son preparados más no inyectados al contexto de filtro o visual, esto quiere decir que no se aplican al modelo de datos o entramado aún, tan sólo se construyeron respetando el contexto de evaluación original, por eso se dice que los filtros quedan latentes.

ii. Se crea una copia del contexto de filtro original, sin ningún contexto de fila, dado que en el siguiente paso *(si aplica)* serán invalidados por la transición de contextos *(entonces, por optimización no vale la pena copiarlos).*

> **Propiedad Conmutativa**: Los pasos 1 y 2 son "intercambiables", empero, el orden exacto del algoritmo es el presentado aquí.

iii. Se ejecuta la operación de transición de contextos si previamente se ha generado un contexto de fila bien sea automático o programable, con lo cual se invalidan los contextos de fila, y de allí, todos las columnas actuales y sus valores son creados como parámetros de filtros implícitos e inyectados a la copia del contexto de filtro.

> Es posible evitar la imposición de sobreescritura de filtros añadidos por transición de contextos (argumentos implícitos o fantasmas), encerrando la expresión tabular donde se itera con KEEPFILTERS (especialmente útil para mantener la forma de filtro arbitrario)

iv. Se ejecutan los parámetros que actúen como modificadores de CALCULATE: USERELATIONSHIP y CROSSFILTER; ALL, ALLEXCEPT, ALLNOBLANKROW, ALLCROSSFILTERED, REMOVEFILTERS y ALLSELECETED; EXPAND, EXPANDALL, COLLPASE y COLLPASEALL.

> Gracias a este orden, se puede anular la transición de contextos con cualquier función ALLxxxx. (Especialmente útil para columnas calculadas)

v. CALCULATE inyecta todos los parámetros de filtros explícitos a la copia del contexto del filtro, haciendo que convivan e intersequen aquellos que afecten la misma columna y vayan encerrados en KEEPFILTERS.

vi. Se aplican todos los filtros del nuevo y más reciente contexto de filtro a el modelo de datos: a todas las tablas que corresponda, generando un nuevo subconjunto de datos visibles (contexto de evaluación).

vii. Se ejecuta el primer parámetro: *Expresión escalar,* en el nuevo y más reciente contexto de evaluación.

viii. Una vez finalizado el más reciente contexto de evaluación es descartado regresando al contexto original.

Por ejemplo:

Acerca de las Cartas DAX

Las Cartas DAX del equipo de **Power Elite Studio LLC & SAS** es un paquete de contenido de documentación de todas las funciones en Leguaje DAX.

- **Incluye:**

Página web, Ficha técnica y Archivos de ejemplos.

→ www.CartasDax.Com ←

Recursos de Aprendizaje del Lenguaje DAX:

MAGÍSTER EN LEGUAJE DAX

Curso Pre-Grabado Completo:

→ [Curso Aquí] ←

bit.ly/3xB5WcD

Última Actualización:

25 de septiembre del 2024

```dax
IngDeDiaMayorParaColSegunLlegada =
MAXX (
    -- (2) Ejecución de transición de contextos:
    Calendario ,
    CALCULATE (
        -- (5) Evaluación de la expresión:
        [Ingresos Tot] ,
        -- (3) Ejecución de modificador:
        USERELATIONSHIP ( Pedidos[FechaLlegada] , Calendario[Fecha] ) ,
        -- (1) Preparación de filtro (queda latente):
        -- (4) Inyección de filtro al contexto de filtro nuevo:
        Pedidos[País] = "Colombia"
    )
)
```

Véase que el orden es totalmente diferente a una secuencia lógica que dicta la intuición, puesto que, el primer parámetro de CALCULATE es el último en ejecutarse.

BIBLIOGRAFÍA

Curso-Capacitación:

- POWER ELITE STUDIO: https://powerelite.studio/cursos/magister-en-lenguaje-dax/

Páginas Web:

- DAX GUIDE: https://dax.guide/calculate/
- MICROSOFT: https://docs.microsoft.com/en-us/dax/calculate-function-dax
- SQL BI: https://www.sqlbi.com/articles/filter-arguments-in-calculate/
- OFFICE: https://support.office.com/en-us/article/context-in-dax

Libros:

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

Creado por:
Miguel Caballero Sierra
Contribución de:
Manuel Ramón Ramón
Cualquier Retroalimentación: powerelitestudio@gmail.com

Funciones Relacionadas: CALCULATETABLE