

Función SUMX

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
Escalar

Recursos de Aprendizaje



MAGÍSTER EN LEGUAJE DAX

→ [Capacitación OnLine] ←

<https://bit.ly/3bzlkG0>

 SUMX: Grifo de la Iteración Agregada

Descripción

La función **SUMX** realiza la suma de todos los valores parciales que «salieron» como resultado de una expresión que se evalúa fila a fila en una tabla.

Sintaxis



SUMX (< Tabla > , < Expresión >)

1 Tabla

Puede ser algo tan sencillo como el nombre de una tabla, o algo más elaborado mediante una expresión de tipo tabla.

 Iterador  Expresión Tabular

Tipo

Obligatorio

Atributo

No Repetible 

2 Expresión

Una expresión: aritmética o algebraica para ser evaluada en una base fila a fila.

 Contexto de Fila

Tipo

Obligatorio

Atributo

No Repetible 

← Valor Que Retorna →

Retorna la suma de todos los valores calculados individualmente fila por fila en la tabla, el resultado puede ser de cualquier tipo.

OBSERVACIONES

1. El primer parámetro es versátil, puesto que, como primera posibilidad puede aceptar simplemente la llamada de una tabla en el modelo, ejemplo 1:

1. IngresosSUMX =
2. SUMX (Pedidos , Pedidos[Ingresos])



Otros Recursos de Aprendizaje



El ADN de Power Pivot

Capítulo número 6

→ [Visitar Libro] ←

El libro **El ADN de Power Pivot** es un manuscrito para estudiar los fundamentos y aspectos intermedios del lenguaje DAX, utilizando Excel y específicamente Power Pivot como herramienta cliente, para la creación de expresiones DAX cuyas soluciones se proyectan a través de tablas dinámicas, además, materializando tablas en la hoja de cálculo.

Es una primera guía para adentrarse en el mundo de DAX, sin embargo, si se requiere más información de la función TREATAS se debe optar por un video curso como el [Máster en DAX y Power Pivot](#) o capacitaciones online y 100% en vivo como: [Énfasis en Contexto de Filtro](#).

→ <http://eladndepowerpivot.com/> ←

Como segunda posibilidad es licito indicar cosas más interesantes mediante expresiones tabulares.

• Ejemplo 2 — Expresión Tabular con [FILTER](#)

Si necesitamos elaborar una expresión que devuelve el valor de los ingresos, pero sin tener en cuenta fines de semana, podemos sostenernos en el hecho de que en el primer parámetro de [SUMX](#) podemos suministrar una tabla que ya está restringida sólo para los días de *lunes* a *viernes*, con la cual, la siguiente expresión brinda una opción de solución:

```

1. IngresosDiasLaborales =
2. SUMX (
3.     FILTER (
4.         Pedidos ,
5.         NOT ( RELATED ( Calendario[DS Número] ) IN { 6 , 7 } ) ) ,
6.     Pedidos[Ingresos]
7. ) -- Véase que el operador NOTIN no existe en DAX, sin embargo,
8.    -- Con invertir el resultado con NOT emulamos algo similar.
    
```

La expresión de tipo tabla del primer parámetro puede ser tan compleja como necesitemos.

• Ejemplo 3 — Expresión de Tipo Tabla con [ADDCOLUMNS](#)

Otro ejemplo, los ingresos aplicando los descuentos almacenados en una tabla de dimensión:

```

1. IngresosConDescuento =
2. SUMX (
3.     ADDCOLUMNS (
4.         Pedidos,
5.         "@Ingresos",
6.         [Ingresos Tot] * ( 1 - RELATED ( Descuentos[Descuento] ) ) )
7.     ),
8.     [@Ingresos]
9. )
10. -- La solución presentada es ilustrativa, ya que hay otras formas.
    
```

II. El segundo parámetro indica una expresión a ser ejecutada fila a fila.

Esta expresión puede ser algo tan sencillo como llamar a un campo en el modelo de datos:

- Ejemplo 4:

```
1. IngresosSUMX =
2. SUMX ( Pedidos , Pedidos[Ingresos] )
```



La medida anterior es la versión larga de: **SUM(Pedidos[Ingresos])**, es decir, esta última es una syntax sugar de la expresión más larga y completa: **SUMX (Pedidos , Pedidos[Ingresos])**, por lo anterior, vemos que cobra mayor utilidad implementar la función **SUMX** cuando el segundo parámetro involucra varias columnas a operar, por ejemplo:

- Ejemplo 5:

```
1. CostoTotal =
2. SUMX (
3.     Pedidos ,
4.     Pedidos[Costo Producto] +
5.     Pedidos[Costo Empaque] +
6.     Pedidos[Costo Envió]
7. )
```

Incluso podemos involucrar funciones, ejemplo 6:

```
1. CostoTotalComoMáximoEntero =
2. SUMX (
3.     Pedidos ,
4.     ROUNDUP ( Pedidos[Costo Producto] , 0 ) +
5.     ROUNDUP ( Pedidos[Costo Empaque] , 0 ) +
6.     ROUNDUP ( Pedidos[Costo Envió] , 0 )
7. )
```

Practica Radiactiva



¡Advertencia! Lo comentado en la parte derecha NO indica que llamara a medidas sea una mala práctica, todo lo contrario, es un buen hábito; no obstante, la precaución se debe tener cuando se hace referencia a medidas en el parámetro expresión de los iteradores (como SUMX), a razón de que, si no es necesario incurrimos unas operaciones internas innecesarias y posibles imprecisiones. Eso sí, la implementación de medidas en el segundo parámetro de iteradores tiene sus aplicaciones importantes, por ejemplo, cálculos en tamaño de granos más grueso, medidas semi aditivas, etc.

Es posible llamar o hacer referencia a medidas en el segundo parámetro de **SUMX**, por ejemplo:

```
1. =
2. SUMX (
3.     Pedidos,
4.     [Ingresos Tot] - [Costos Tot]
5. )
```

Empero, si el cálculo requerido se puede resolver llamando a las columnas involucrados, directamente, es decir, así:

```
1. =
2. SUMX (
3.     Pedidos ,
4.     Pedidos[Ingresos] - Pedidos[Costo Total]
5. )
```

Entonces, crearlo llamando a columnas, ya que al hacer referencia a medidas en el segundo parámetro de **SUMX** y en general, de cualquier iterador, provoca la activación de la *transición de contextos*, la cual es una operación de alto costo y si contamos con filas duplicadas en ausencia de una columna de valores únicos o que no se ha marcado como tal, entonces, presentará valores imprecisos al inflarlos, en la mayoría casos de forma muy difícil de detectar, puesto que dicha inflación es minúscula. [Más Detalles Aquí](#)

III. La función **SUMX** ignora la existencia de valores *null*, tomados como vacío (**BLANK**) por el motor DAX, realizando la determinación de valor superior exclusivamente sobre el conjunto de valores estricta y visualmente numéricos.

Por esta consideración, si la distribución de la columna o del resultado de la expresión ejecutada fila a fila tiene valores vacíos que deben ser leídos como **0**, entonces, se debe asignar el valor explícitamente.

- Ejemplo:



Acerca de las Cartas DAX



Las cartas DAX del equipo de **Excel Free Blog** es un paquete de contenido de documentación y representación para un juego de todas las funciones en lenguaje 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:
 20 de mayo del 2021.

IV. La función SUMX no admite valores de tipo TRUE/FALSE en el resultado de la expresión ejecutada fila a fila, por lo tanto, aplicar lógica booleana de forma directa para resolver el ejemplo anterior no es valido

• Ejemplo: 7

```
1. SumaERROR :=
2. SUMX (
3.     Pedidos;
4.     NOT ISBLANK ( Pedidos[Ingresos] ) * VALUE ( Pedidos[Ingresos] )
5. ) -- Error: La función no admite valores booleanos
```

Aunque con funciones como **CONVERT** se puede lograr, esto hace más enrevesado e ineficiente el proceso.

V. Si la expresión pasada en su primer parámetro retorna una tabla vacía (no hay filas) entonces la función **SUMX** devuelve **BLANK**.

Dado que para una visualización el contexto vario, el valor **BLANK** puede aparecer en casillas o lugares específicos del objeto visual.

REFERENCIAS Y BIBLIOGRAFÍA

Páginas Web:

- 1. DAX GUIDE: <https://dax.guide/maxx/>
- 2. MICROSOFT: <https://docs.microsoft.com/en-us/dax/maxx-function-dax>
- 3. EFB: <https://www.excelfreeblog.com/trampa-de-sum-implementada-en-sumx/>
- 4. EFB: <https://www.excelfreeblog.com/principios-en-funciones-de-iteracion-escalares>

Creado por:

Miguel Caballero y Fabian Torres.

Cualquier Retroalimentación:

excelfreebymcs@gmail.com

Funciones Relacionadas:  **SUM**



SUMX: Grifo de la Iteración Agregada