Función ADDCOLUMNS

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
Genera un contexto de fila programable

Categorías

Según Proceso Interno
Iteración

★★★★★

Según Resultado
Tabla

Recursos de Aprendizaje

ADDCOLUMNS: Rugido de Columnas

Descripción

La función ADDCOLUMNS retorna una tabla o expresión de tabla con nuevas columnas añadidas de carácter temporal, visto de otro modo, devuelve una tabla con columnas calculadas, esto quiere decir que se crean mediante una expresión que se determina en una base fila a fila.

Sintaxis

1 Primer Parámetro
2 Segundo Parámetro
Tercer Parámetro 3

```
ADDCOLUMNS ( < Tabla > , < Nombre > , < Expresión>
                [ , < Nombre > , < Expresión> ] [, ...] ] )
```

4 Cuarto Parámetro (Opcional)
Quinto Parámetro (Opcional) 5

1 Tabla

La expresión de tabla a la cual se le agregaran nuevas columnas.

ⓘ Iterador ⓘ Expresión de Tabla

Tipo
Obligatorio

Atributo
No Repetible

2 Nombre

Nombre de la nueva columna. *(Es importante seguir alguna convención para distinguir de las medidas).* Al ser una columna temporal puede tener un nuevo data lineage *(Véase la Función TREATAS).*

Tipo
Obligatorio

Atributo
Repetible

3 Expresión

Expresión de tipo escalar al ser evaluada fila a fila; de allí se construye los valores de la columna.

ⓘ Contexto de Fila

Tipo
Obligatorio

Atributo
Repetible

Más Recursos de Aprendizaje

Máster en DAX y Power Pivot

Tomo Número 7: Módulo 22

→ [Visitar Curso] ←

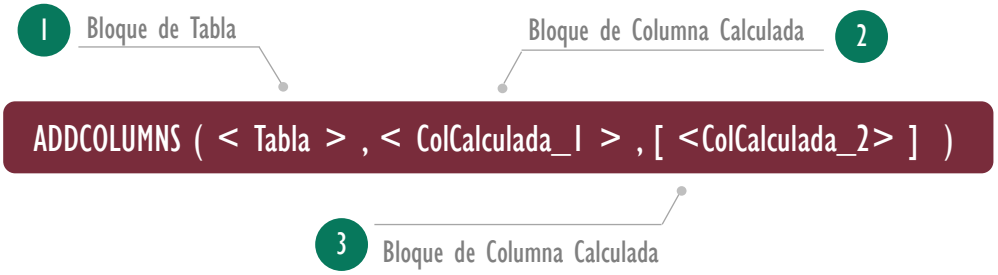
El curso **Máster en DAX y Power Pivot es un vídeo curso pregrabado de 24 horas** dedicado a revelar y explorar todos los secretos del lenguaje DAX y la interfaz de Power Pivot en Excel.

A lo largo de sus 164 lecciones distribuidas en 22 módulos **se abraza las mieles del análisis de datos para disfrutar de las bondades de la creación de KPIs y KRIs** de todo tipo utilizando las funciones DAX y tablas dinámicas, cada archivo cuenta con la documentación completa del material tratado con expresiones DAX y explicación.

→http://bit.ly/2TUPoUF←

La sintaxis de la función ADDCOLUMNS se puede agrupar en dos bloques esenciales de parámetros: *Bloque de Tabla* y *Bloque de Columnas Calculadas*, así:

Sintaxis Agrupada



La razón para mostrar la sintaxis de forma agrupada estriba en dos pilares:

i. Nos hacemos consientes que del segundo parámetros en adelante trabajan en parejas de parámetros: (<Nombre> , <Expresión>)

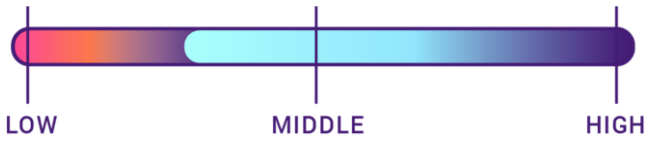
ii. Además, para todos los fines prácticos: son columnas calculadas creadas temporalmente y vienen escoltadas de las ventajas y desventajas de sus primas físicas.

Algunas consideraciones primordiales:

Almacenadas en memoria, por lo que: consumen memoria RAM.

Fáciles se de utilizar e inofensivas para cardinalidad de columnas baja.

Condición para un filtro en un a consulta o creación de categorías.



Recalculadas en el momento de actualización del reporte.

Agregación de datos de otras columnas en el modelo y de normalización de tablas.

Cálculos determinados en una base fila a fila y deben estar restringidos.

Otros Recursos de Aprendizaje

En el vídeo de YouTube **Miguel Caballero** proporciona un vistazo a la función **ADDCOLUMNS, pero implementada en consultas para Power Pivot en Excel**.

Además, en el vídeo se presenta una manera de cumplir la convención de escritura entre: *columna calculada* y *medidas* par a columnas temporales, sin embargo, aunque es enfoque válido, el equipo de Excel Free Blog actualmente se ciñe a la convención: @ (que tienes sus excepciones, como lo es el caso para consultas).

→ https://bit.ly/3jjbJq4 ←

Consideraciones Generales

I. **Transición de Contextos:** Al ser una función de iteración genera un contexto de fila explícito *(conocido también como programable)*, quiere decir que para los argumentos expresión, quienes construyen los valores de las columnas, es necesario encerrarlos en CALCULATE para que la transición de contextos se active.

- Ejemplo No.1:

Para una tabla denominada *Pedidos* que registras las ventas diarias, y que además señala el país de la transacción, podemos obtener el ingreso generado por cada país en una tabla calculada, así:

```
IngPorPais =
ADDCOLUMNS (
    VALUES ( Pedidos[País] ),
    "Ingresos",
        CALCULATE (
            SUM ( Pedidos[Ingresos] ) -- Es necesario CALCULATE.
        )
)
-- La expresión es para una tabla calculada.
```

CALCULATE IMPLÍCITO

El CALCULATE en los argumentos expresión no es necesario si previamente tenemos la medida creada, con la cual podemos referenciarla directamente y apoyarnos en el hecho de que, al llamar una medida, esta automáticamente queda encerrada en un CALCULATE implico.

Si tenemos la medida [Ingresos Tot] que se define así: *SUM(Pedidos[Ingresos]), entonces, la expresión se pude escribir:*

```
IngPorPais =
ADDCOLUMNS (
    VALUES ( Pedidos[País] ),
    "Ingresos",
    [IngresosTot]
)
```

Cualquier expresión o función de tabla, por sencilla o compleja que sea es válida en el primer argumento de ADDCOLUMNS.

- Ejemplo No. 2:

Si desea mostrar los ingresos por país y por año, se puede solucionar la tabla calculada con ayuda de la función SUMMARIZE, así:

```
IngPorPais =
ADDCOLUMNS (
    SUMMARIZE (
        Pedidos,
        Calendario[Año],
        Pedidos[País]
    ),
    "Ingresos",
        CALCULATE (
            SUM ( Pedidos[Ingresos] )
        )
)
```

La creación de columnas temporales lo puede realizar la función SUMMARIZE por si sola sin necesidad de ADDCOLUMNS, sin embargo, esta operación con SUMMARIZE se considera obsoleta por su ineficiencia y algoritmo complejo, lo recomendable es lo presentado en la presente ficha técnica. ***Más detalles sobre SUMMARIZE aquí.***

ADDCOLUMNS Y CALENDARIO

La función ADDCOLUMNS también es ampliamente implementada para la construcción de tablas De Calendario, también conocidas como tablas de fechas.

Dado que con ella en una sola expresión se puede construir toda la tabla.

A pesar de que, en la presente ficha técnica no se deja la expresión, en el siguiente vídeo se puede estudiar en detalle:

CONSTRUCCIÓN DE TABLA DE CALENDARIO

A PARTIR DEL MINUTO: 50:51

Allí mismo se puede estudiar la construcción de l tabla de Calendario con la función GENERATE y Variables.

- Ejemplo No. 3:

Otro ejemplo, desde la perspectiva de tablas calculadas o consultas, los ingresos sólo para los días laborales, teniendo una tabla de fechas festivas.

```
IngDiasLaborales =
VAR SinFestivos =
    EXCEPT (
        VALUES ( Calendario[Fecha] ) ,
        VALUES ( FechasFestivas[Fecha de Festividad] )
    )
VAR SinFestivosyFD =
    FILTER (
        SinFestivos,
        WEEKDAY (
            Calendario[Fecha],
            1
        ) < 6
    )
VAR IngresosDiasLaborales =
    ADDCOLUMNS (
        SinFestivosyFD;
        "Ingresos", [IngresosTot]
    )
RETURN
    IngresosDiasLaborales
```

II. **Convención para Nombrar Columnas:** ADDCOLUMNS no se limita a tablas calculadas y consultas, también es implementada en medidas, sin embargo, al ser utilizada en este ámbito es buena practica tener una convención para los nombres de las columnas generadas, dado que son columnas que no son parte de ninguna tabla nativa, por lo tanto, no se puede indicar la tabla en la que reside, esto deriva en que la convención para distinguir entre columnas calculada y medidas se rompe.

CONVENCIÓN: @

El equipo de SQLBI ha propuesto **poner cómo prefijo el símbolo: @ para de esa forma distinguir las columnas temporales**, teniendo en cuenta que es un carácter con una probabilidad baja de implementarse en nombres de columnas, tablas, medidas y demás.

Es importarte remarcar que **esta convención se omití si el resultado se va a materializar**, es decir, si es implementado en una consulta o tabal calculada, dado que el @ se propagaría y no sería amigable, por ello, en los ejemplos anteriores no se implemento la convención, ya que **el equipo de Excel Free Blog adopta y recomienda esta buena práctica**.

Sobre Tabla de Hecho y Granularidad:

https://bit.ly/3jy2qTj

- Ejemplo con Medidas No. 1:

Una manera de implementar ADDCOLUMNS en medida es para cálculo en tamaño de grano más grueso.

Para entenderlo visualicemos la tabla *Pedidos* donde tenemos los *registros día a día.* Ahora bien, si necesitamos presentar en una matriz con los años en filas el *ingreso promedio*, se logra fácilmente con la siguiente medida:

```
1. Promedio Ingresos =
2. AVERAGE ( Pedidos[Ingresos] )
```

Pero, no debemos perder de vista que dicha medida en realidad se lee:

Promedio de los ingresos a nivel de día según el contexto.

Es decir, se lee el detalle más fino que nos proporciona la tabla *Pedidos*, por lo tanto, lo que se lee es:

Empero un requisito común consiste en: *obtener el promedio de ingresos a nivel de mes manteniendo la configuración de matriz inicial.*

Enfoque 1

El enfoque de AVERAGEX con ADDCOLUMNS tiene la ventaja que es un método que se puede aplicar a múltiples atributos.

Ejemplo, el promedio de los ingresos a niel de Mes y a nivel de SKU.

Para el caso señalado se aplica la combinación ADDCOLUMNS con SUMMARIZE como en los ejemplos iniciales y se itera con AVERAGEX.

Se puede argüir que en lugar de SUMMARIZE, se puede optar por: GROUPBY y hasta SUMMARIZECLUMNS.

Para más información, se recomienda la siguiente lectura:

https://www.excelfreeblog.com/summarize-vs-groupby-vs-summarizecolumns-2/

Un enfoque de solución a la medida demandada:

```
IngPromedioNMes =
AVERAGEX (
    ADDCOLUMNS (
        VALUES ( Calendario[Fecha] ),
        "@Ingresos",
            CALCULATE (
                SUM ( Pedidos[Ingresos] )
            )
    ),
    [@Ingresos] -- El color de la columna temporal es una ayuda
)               -- Visual para la ficha técnica y distinguir de
                -- Como columna temporal.
```

La expresión anterior se puede escribir de manera más compacta y sin la necesidad de ADDCOLUMNS, así:

```
IngPromedioNMes =
AVERAGEX (
    Calendario,
    [Ingresos Tot]
)
```

Acerca de los Dos Enfoques

Dependiendo del escenario, distribución de datos, relaciones. Etc, El rendimiento de una u otra puede ser mejor.

La implementación de ADDCOLUMNS en medida también se presenta en optimización de cálculos, específicamente, en las medias semi aditivas que generalmente requieren iteraciones anidadas.

- Ejemplo con Medidas No. 2:

Una medida que retorne el último balance dado que los clientes tienen fechas distintas de registro final, por consiguiente, su balance es el último dato en el periodo, mientras que el balance total del *"banco"* es la sumatoria de los últimos balances de cada cliente:

```
ÚltimoBalanceOptimizado =
VAR UltimoClienteyFecha =
    ADDCOLUMNS (
        VALUES ( Balance[Nombre] ),
        "UltimaFecha", CALCULATE (
            MAX ( Balance[Fecha] ),
            DATESBETWEEN (  Calendario[Fecha],
                            BLANK (),
                            MAX ( Calendario[Fecha] )
             )
         )
    )
VAR FiltrosDeCleintesyFechas =
    TREATAS ( UltimoClienteyFecha, Balance[Nombre], Calendario[Fecha] )
VAR SumaDelUltimoBalance =
    CALCULATE ( SUM ( Balance[Balance] ), FiltrosDeCleintesyFechas )
RETURN
    SumaDelUltimoBalance
```

Acerca de las Cartas DAX

Las cartas DAX del equipo de **Excel Free Blog** es un paquete de contenido de documentación y representación para 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:
21 de febrero del 202

ADDCOLUMNS: Rugido de Columnas

BIBLIOGRAFÍA

Páginas Web:

- 1. DAX GUIDE: https://dax.guide/addcolumns/
- 2. MICROSOFT: https://docs.microsoft.com/en-us/dax/addcolumns-function-dax
- 3. SQLBI: https://www.sqlbi.com/articles/best-practices-using-summarize-and-addcolumns/
- 4. SQLBI: https://www.sqlbi.com/articles/naming-temporary-columns-in-dax/
- 5. SQLBI: https://www.sqlbi.com/articles/all-the-secrets-of-summarize/
- 6. Excel Free Blog: https://www.youtube.com/watch?v=kZfz9QUvSI0&t=1055s
- 7. Excel Free Blog: https://www.youtube.com/watch?v=u4WyLfyflJE&t=3045s
- 8. Excel Free Blog: https://www.excelfreeblog.com/summarize-vs-groupby-vs-summarizecolumns-2/
- 9. Wikipedia: https://es.wikipedia.org/wiki/Tabla_de_hechos

Libros:

- Definitive Guide To DAX (2nd Edition) – Marco Russo y Alberto Ferrari
- Exam Ref 70-778 Analyzing and Visualazing Data – Daniil Maslyuk

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

Funciones Relacionadas: SELECTCOLUMNS