DAX: Variabelen gebruiken om uw formules te verbeteren

5-2-2020 • 5 minutes to read • Edit Online

Als gegevensmodelbouwer kan het schrijven en opsporen van fouten in sommige DAX-berekeningen een ware uitdaging zijn. Het komt vaak voor dat voor complexe berekeningsvereisten samengestelde of ingewikkelde expressies moeten worden geschreven. Voor samengestelde expressies moeten mogelijk geneste functies worden gebruikt. Mogelijk moet expressielogica worden hergebruikt.

Door variabelen in uw DAX-formules te gebruiken, kunt u complexe en efficiënte berekeningen schrijven. Met variabelen kunt u:

- De prestaties verbeteren
- De leesbaarheid verbeteren
- Foutopsporing vereenvoudigen
- De complexiteit verkleinen

In dit artikel laten we u de eerste drie voordelen zien aan de hand van een voorbeeldmeting voor de jaarlijkse verkoopgroei (YoY-verkoopgroei). (De formule voor de YoY-verkoopgroei is als volgt: verkoop in periode _ verminderde verkoop voor dezelfde periode vorig jaar, *gedeeld door* de verkoop voor dezelfde periode vorig jaar.)

We beginnen met de volgende metingsdefinitie.

```
Sales YoY Growth % =
DIVIDE(
    ([Sales] - CALCULATE([Sales], PARALLELPERIOD('Date'[Date], -12, MONTH))),
    CALCULATE([Sales], PARALLELPERIOD('Date'[Date], -12, MONTH))
)
```

Met de meting wordt weliswaar het juiste resultaat verkregen, maar de definitie is voor verbetering vatbaar.

De prestaties verbeteren

U ziet dat de expressie waarmee 'dezelfde periode vorig jaar' wordt berekend, in de formule wordt herhaald. Dit is een vrij inefficiënte formule omdat Power BI hiervoor twee keer dezelfde expressie moet evalueren. U kunt de metingsdefinitie efficiënter maken door een variabele te gebruiken.

De volgende definitie van een meting is al een verbetering. Er wordt een expressie gebruikt om het resultaat van 'dezelfde periode vorig jaar' aan de variabele **SalesPriorYear** toe te wijzen. Vervolgens wordt deze variabele twee keer gebruikt in de RETURN-expressie.

```
Sales YoY Growth % =
VAR SalesPriorYear =
    CALCULATE([Sales], PARALLELPERIOD('Date'[Date], -12, MONTH))
RETURN
    DIVIDE(([Sales] - SalesPriorYear), SalesPriorYear)
```

De meting levert nog steeds het juiste resultaat op, maar heeft hiervoor nu maar de helft van de querytijd nodig.

De leesbaarheid verbeteren

U zag in de vorige metingsdefinitie dat de RETURN-expressie dankzij de gekozen naam voor de variabele eenvoudiger te begrijpen is. De expressie is kort en zelf-beschrijvend.

Foutopsporing vereenvoudigen

U kunt variabelen ook gebruiken om fouten in een formule op te sporen. Voor het testen van een expressie die aan een variabele is toegewezen, moet u de RETURN-expressie tijdelijk herschrijven om de variabele uit te voeren.

Met de volgende metingsdefinitie wordt alleen de variabele **SalesPriorYear** geretourneerd. U ziet dat de opmerkingen bij de bedoelde RETURN-expressie worden verwijderd. Door deze techniek kunt u eenvoudig terugzetten zodra de foutopsporing is voltooid.

```
Sales YoY Growth % =
VAR SalesPriorYear =
    CALCULATE([Sales], PARALLELPERIOD('Date'[Date], -12, MONTH))
RETURN
    --DIVIDE(([Sales] - SalesPriorYear), SalesPriorYear)
    SalesPriorYear
```

De complexiteit verkleinen

In eerdere DAX-versies werden variabelen nog niet ondersteund. Er waren complexe expressies met nieuwe filtercontexten vereist om de DAX-functies EARLIER of EARLIEST te gebruiken om naar externe filtercontexten te verwijzen. Helaas vonden gegevensmodelbouwers deze functie moeilijk om te begrijpen en te gebruiken.

Variabelen worden altijd geëvalueerd buiten de filters die door uw RETURN-expressie worden toegepast. Hierdoor krijgt u hetzelfde resultaat als de functie EARLIEST wanneer u een variabele binnen een aangepaste filtercontext gebruikt. De functies EARLIER of EARLIEST hoeven daardoor niet te worden gebruikt. Het betekent dat u nu minder complexe formules kunt schrijven die eenvoudiger te begrijpen zijn.

Bekijk de volgende definitie van een berekende kolom die aan de tabel **Subcategorie** is toegevoegd. Met deze definitie wordt voor elke subcategorie een positie geëvalueerd op basis van de waarden in de kolom **Verkoop in subcategorie**.

```
Subcategory Sales Rank =
COUNTROWS(
    FILTER(
        Subcategory,
        EARLIER(Subcategory[Subcategory Sales]) < Subcategory[Subcategory Sales]
    )
) + 1
```

De functie EARLIER wordt gebruikt om naar de waarde van de kolom **Verkoop in subcategorie***in de huidige rijcontext* te verwijzen.

De definitie voor de berekende kolom kan worden verbeterd met behulp van een variabele in plaats van de functie EARLIER. Met de variabele **CurrentSubcategorySales** wordt de waarde in de kolom **Verkoop in subcategorie***in de huidige rijcontext* opgeslagen. Deze variabele wordt door de RETURN-expressie in een aangepaste filtercontext gebruikt.

```
Subcategory Sales Rank =
VAR CurrentSubcategorySales = Subcategory[Subcategory Sales]
RETURN
    COUNTROWS(
        FILTER(
            Subcategory,
            CurrentSubcategorySales < Subcategory[Subcategory Sales]
        )
    ) + 1
```

Volgende stappen

Bekijk de volgende bronnen voor meer informatie over dit artikel:

- VAR DAX-artikel
- Vragen? Misschien dat de Power BI-community het antwoord weet