

Advanced DAX – Relationship Functions & Parent - Child Function

Physical vs Logical Relationship

Physical Relationship: One created in data model

Logical Relationship: One Created using DAX function

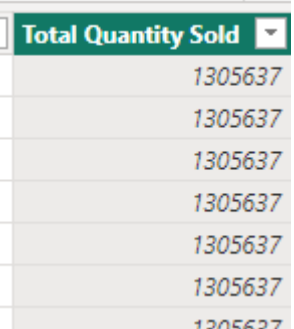
Relationship Functions

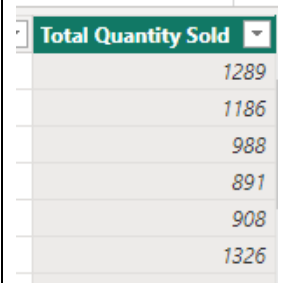
RELATED vs RELATEDTABLE vs CROSSFILTER

RELATED works from the many-side of a relationship towards the one-side. When you need to traverse the relationship in the opposite direction, you can use RELATEDTABLE

e.g. Sales table and product table has many to one relationship flowing from product to Sales. In this case if you want any column from product table (product name) in sales table then RELATED can be used.

Now let's try to do other way round. In below example I am trying to add Total Quantity Sold from Sales table into product lookup table. Let us see how RELATED and RELATEDTABLE results differ from each other.

<pre>Total Quantity Sold = SUM('Sales by Store'[quantity_sold])</pre>	
---	---

<pre>Total Quantity Sold = SUMX(//RELATED('Sales by Store'), --This will not work RELATEDTABLE('Sales by Store'), 'Sales by Store'[quantity_sold])</pre>	
--	--

Other way to do this is enable Cross filtering direction to “Both” in physical relationship or using USERELATIONSHIP function in DAX.

<pre>Total Quantity Sold(Using CROSSFILTER Function) = CALCULATE(SUM('Sales by Store'[quantity_sold]), CROSSFILTER('Sales by Store'[product_id], 'Product Lookup'[product_id], Both))</pre>

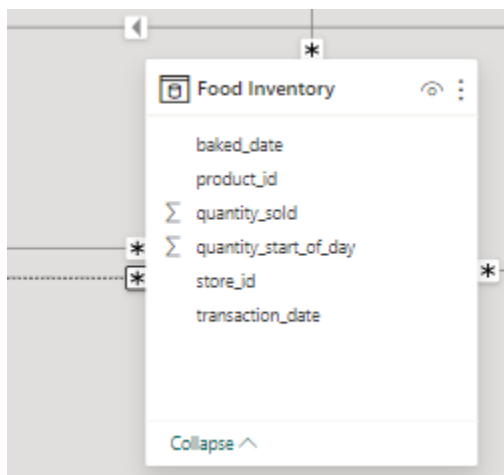
urrent_wholesale_price	current_retail_price	tax_exempt_yn	promo_yn	new_product_yn	Total Quantity Sold	Total Quantity Sold(Using CROSSFILTER Function)
14.4	18	Y	N	N	1289	1289
14.4	18	Y	N	N	1186	1186
11.8	14.75	Y	N	N	988	988
16.36	20.45	Y	N	N	891	891
12	15	Y	N	N	908	908
16.8	21	Y	N	N	1326	1326
15.8	19.75	Y	N	N	926	926
36	45	Y	N	N	1479	1479
18	22.5	Y	N	N	1233	1233
8	10	Y	N	N	869	869
7.16	8.95	Y	N	N	981	981
7.16	8.95	Y	N	N	861	861

USERRELATIONSHIP

USERRELATIONSHIP function is used when there is no active relationship between two columns of different tables. This is useful in case of role-playing dimensions.

Note that to use USERRELATIONSHIP function you need to create an inactive relationship.

In below example, we have two dates in food inventory table – Baked Date and Transaction Date. I have created active relationship on Transaction Date of Inventory table with Calendar table transaction date and one inactive relationship using Baked Date.



```
Sales By Transaction Date (USERRELATIONSHIP) =  
CALCULATE(  
    SUM('Food Inventory'[quantity_sold]),  
    USERRELATIONSHIP(  
        'Sales by Store'[transaction_date],  
        'Calendar'[Transaction_Date]  
    )  
)
```

```
Sales By Baked Date (USERRELATIONSHIP) =  
CALCULATE(  
    SUM('Food Inventory'[quantity_sold]),  
    USERRELATIONSHIP(  
        'Food Inventory'[baked_date],  
        'Calendar'[Transaction_Date]  
    )  
)
```

Transaction_Date	Sales By Transaction Date (USERRELATIONSHIP)	Sales By Baked Date (USERRELATIONSHIP)
2017-01-03	75	85
2017-01-04	72	71
2017-01-05	76	70
2017-01-06	82	102
2017-01-07	86	111
2017-01-08	102	96
2017-01-09	73	122
2017-01-10	104	89
2017-01-11	92	118
2017-01-12	97	78
2017-01-13	101	69
2017-01-14	112	70
2017-01-15	101	114
2017-01-16	100	112
2017-01-17	75	97
2017-01-18	82	112
2017-01-19	90	83
2017-01-20	127	92

Parent and Child Functions – Use Cases

PATH

Finding all levels of reporting for an employee in each row

```
Hierarchy Demo (PATH) =  
  PATH (  
    'Employee Hierarchy'[EmployeeKey], 'Employee Hierarchy'[ParentEmployeeKey]  
  )
```

PATHLENGTH

Finding total number of people in your hierarchy till topmost employee

```
Hierarchy Demo (PATHLENGTH) =  
  PATHLENGTH(  
    PATH (  
      'Employee Hierarchy'[EmployeeKey], 'Employee Hierarchy'[ParentEmployeeKey]  
    )  
  )
```

PATHREVERSE

Finding second level of manager for escalation. Note that employee hierarchy returned by PATH function is from Top to bottom hence Reversing string is necessary to find second level of manager

```
Second Level Manager =  
  PATHITEMREVERSE(  
    PATH('Employee Hierarchy'[EmployeeKey], 'Employee Hierarchy'[ParentEmployeeKey]),  
    3, //Returns Third element in path once it is reversed  
    1  
  )
```

PATHITEM

```
Business Head(PATHITEM) =  
  
PATHITEM(  
    PATH('Employee Hierarchy'[EmployeeKey], 'Employee  
Hierarchy'[ParentEmployeeKey]),  
    2, //Returns Second element (Reporting to CEO) in path  
    1  
)
```

PATHCONTAINS

```
Reporting To "Peter Krebs" =  
IF(  
    PATHCONTAINS(  
        PATH('Employee Hierarchy'[EmployeeKey], 'Employee Hierarchy'[ParentEmployeeKey]),  
        "23"  
    ),  
    "Yes",  
    "No"  
)
```

Results are all above DAX formulae are shown in below screenshots. All are added as calculated columns in Employee Hierarchy table

Empl	ParentEmpl	FullName	Hierarchy Demo (PATH)	Hierarchy Demo (PATHL)	Second Level Manager	Business Head(PATHITEM)	Reporting To "Peter Krebs"
1	18	Guy Gilbert	112 23 18 1	4	23	23	Yes
2	7	Kevin Brown	112 7 2	3	112	7	No
3	14	Roberto Tamburelli	112 14 3	3	112	14	No
4	3	Rob Walters	112 14 3 4	4	14	14	No
5	3	Rob Walters	112 14 3 5	4	14	14	No
6	267	Thierry D'Hers	112 14 3 267 6	5	3	14	No
7	112	David Bradley	112 7	2		7	No
8	112	David Bradley	112 8	2		8	No
9	23	Jolynn Dobney	112 23 9	3	112	23	Yes
10	189	Ruth Ellerbrock	112 23 189 10	4	23	23	Yes
11	3	Gail Erickson	112 14 3 11	4	14	14	No
12	189	Barry Johnson	112 23 189 12	4	23	23	Yes
13	3	Jossef Goldberg	112 14 3 13	4	14	14	No
14	112	Terri Duffy	112 14	2		14	No
15	189	Sidney Higa	112 23 189 15	4	23	23	Yes
16	23	Taylor Maxwell	112 23 16	3	112	23	Yes
17	189	Jeffrey Ford	112 23 189 17	4	23	23	Yes
18	23	Jo Brown	112 23 18	3	112	23	Yes

Employee Hierarchy

☐ Business Head(PATHITEM)

☐ EmployeeKey

☐ FullName

☐ Hierarchy Demo (PATH)

☐ Hierarchy Demo (PATHLENGTH)

☐ ParentEmployeeKey

☐ Reporting To "Peter Krebs"

☐ Second Level Manager