Lesson 5

Relating Data

If you are a heavy Excel user, you've probably used its omnipresent VLOOKUP function to look up values from another cell or to aggregate data in a range. This is a common task for data modeling too, although you use relationships and tables as opposed to cells and ranges. This lesson teaches you how to navigate tables whether physical relationships exist or not. You'll find the DAX formulas for this lesson in \Source\Part2\Relating Data.dax.

5.1 Navigating Existing Relationships

Recall from Lesson 2 that relationships are very important to Power BI data models. They promote self-service data exploration without requiring you to create queries that join tables. If a relationship exists between a dimension table and a fact table, you can slice and dice the fact data by any field in the dimension table. Calculated columns can benefit from existing relationships too to let you "look up" or aggregate values from a related table. DAX has two functions, RELATED and RELATEDTABLE, for navigating active relationships.

5.1.1 Navigating Many-to-One Relationships

Suppose you want to calculate the net profit for each row in the FactResellerSales table. For our purposes, you'll calculate the line net profit by subtracting the product cost from the line item total. As a first step, you need to look up the product cost in the DimProduct table. In other words, for each row (line item) in FactResellerSales table, you need the product identifier (ProductKey column), then follow the relationship to DimProduct, and look up the value in DimProduct[StandardCost]. This is a many-to-one relationship from the FactResellerSales table perspective (notice the * symbol in **Figure 5.1**)

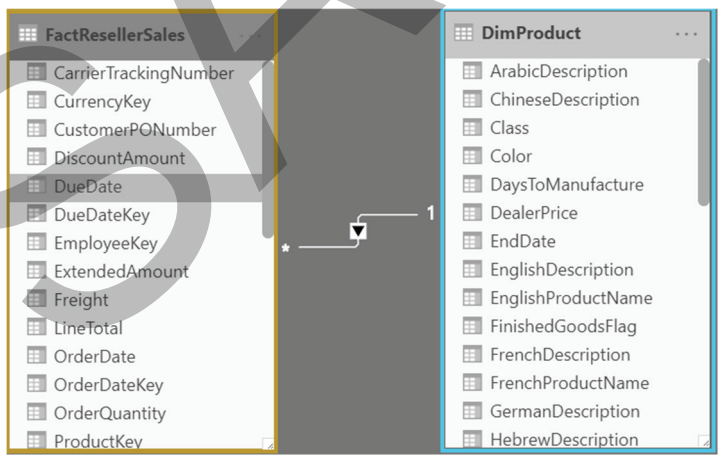

Figure 5.1 Looking up the product cost in DimProduct requires navigating the many-to-one relationship for each row in FactResellerSales.

Practice

When working on a complicated formula, consider breaking it up into multiple steps. Let's focus on looking up the StandardCost first.

1. Make sure the Data or Report View tab is selected in the navigation pane. In the Fields pane, right-click the FactResellerSales table and then click "New column".
2. In the formula bar, enter the following formula and press Enter:

```
NetProfit = RELATED(DimProduct[StandardCost])
```

Output

Power BI adds a new NetProfit column to DimProduct and populates it with standard cost for that product that is recorded in the DimProduct[StandardCost] column.

Analysis

This expression uses the RELATED function to look up the value of the StandardCost column in the Product table. Since a calculated column inherits the current row context, this expression is evaluated for each row. Unfortunately, Power BI doesn't automatically propagate the row context to related tables. Hence, you must use RELATED which is designed to follow a many-to-one relationship and to apply the row context. RELATED has the following definition:

```
Related(<column>)
```

For each row in FactResellerSales, Power BI constructs a row context consisting of all column values in that row, including the ProductKey value. Then, it navigates the ResellerSales[ProductKey] ⇨ Product[ProductKey] relationship, and retrieves the standard cost for that product from the Product[StandardCost] column.

NOTE RELATED requires a row context and an active relationship to the table where the column is located. If there is no active relationship, RELATED returns an error.

To recap, RELATED needs a row context and can be used only in two cases:

- A calculated column expression.
- In an extended "X" function, such as SUMX, that iterates over a table and creates a row context for each row.

Practice

Complete the NetProfit formula to calculate the line net profit:

1. Change the formula as follows;

```
NetProfit = [LineTotal] - (RELATED(DimProduct[StandardCost]) * FactResellerSales[OrderQuantity])
```

2. In the Fields list, select FactResellerSales[NetProfit]. In the Modeling ribbon (Formatting group), format the column values without decimal places.
3. (Optional) Test the NetProfit column, such as to aggregate it by DimDate[CalendarYear], or refer to the "RELATED Function" visual on the page "Relating Data" in \Source\Part2\Adventure Works.pbix, which is shown in **Figure 5.2**.

Analysis

While browsing the values in the NetProfit column in the Data View tab, notice that when the line item's product cost exceeds the line total, the result is a negative value. This is the expected result so don't be alarmed.

CalendarYear	NetProfit
2010	$17,033
2011	$29,420
2012	$915,899
2013	($491,870)
Total	**$470,483**

Figure 5.2 The NetProfit calculated column uses the RELATED function to look up the product standard cost.

Remember that a calculated column can be placed in any area of the report when it makes sense. In this case, NetProfit is a numeric column and you can place it in the visual's Values area to aggregate it. This doesn't make it a measure though. Instead, Power BI creates an implicit measure to summarize the calculated column.

5.1.2 Simplifying the Model Schema

You can use the RELATED function to simplify a snowflake schema by reducing the number of tables. For example, unless there is a good reason to keep these tables separate, you can consolidate DimProduct, DimProductSubcategory and DimProductCategory. While you can accomplish this with custom SQL or Power Query, let's use calculated columns.

TIP Although this is a useful exercise for calculated columns, I recommend you use SQL or Power Query for data shaping. SQL and Power BI are better suited for transformation tasks, such as replacing empty values that don't have a match.

Practice

Denormalizing the product schema with DAX requires two new calculated columns:

1. Add these two columns to DimProduct so that this table has both product category and subcategory:

```
EnglishProductSubcategoryName = RELATED(DimProductSubcategory[EnglishProductSubcategoryName])
EnglishProductCategoryName = RELATED(DimProductCategory[EnglishProductCategoryName])
```

2. (Optional) Hide the DimProductSubcategory and DimProductSubcategory tables. To hide a table, right-click the table in the Fields pane (make sure that the Data View tab or Model View tab are selected) and click "Hide in report view".

Output

The DimProduct table now has the product subcategory and category as calculated columns that derive their values from the related tables. As a result, you managed to collapse three tables into one and thus you simplified the model schema. Notice that the new columns have empty values when there is no match.

Analysis

You can use the RELATED function to navigate many-to-one cascading relationships and look up values from tables that are not directly related to the home table. Think of RELATED as a SQL left join between two tables. In the case where there is no match, it returns an empty value.

5.1.3 Navigating One-to-Many Relationships

Another DAX function, RELATEDTABLE, lets you navigate a relationship in either direction. This is useful when the formula needs to follow a one-to-many relationship, that is from the dimension table to the fact table. It has the following definition:

```
RELATEDTABLE(<tableName>)
```

Like RELATED, RELATEDTABLE propagates the row context. Because it returns a table with a subset of rows that match the current row context, you typically need to aggregate the results if you want to use this function in a calculated column.

NOTE Although less common, Power BI also supports relationships with a many-to-many data cardinality (not to be confused with many-to-many relationships discussed in the lesson "Many-to-many relationships"), such as between two fact tables or between tables in different storage modes (imported and DirectQuery). You can also use RELATEDTABLE to navigate such relationships. For more information about relationships with a many-to-many cardinality, refer to the article "Relationships with a many-many cardinality in Power BI Desktop" at https://docs.microsoft.com/power-bi/desktop-many-to-many-relationships.

Practice

Suppose you need a calculated column in the DimProduct table that summarizes the reseller sales from FactResellerSales for each product: In your first attempt, add the following calculated column to DimProduct:

```
ResellerSales = SUM(FactResellerSales[SalesAmount])
```

Outcome

The DimProduct[ResellerSales] calculated column returns a repeating value that represents the overall sales across the entire FactResellerSales.

Analysis

The formula doesn't propagate the row context of the "current product".

Practice

Change the formula of the calculated column as follows:

```
ResellerSales = SUMX(RELATEDTABLE(FactResellerSales), FactResellerSales[SalesAmount])
```

Outcome

Now the calculated column returns the expected results. Notice that some products, such as accessories, show no sales because they are never sold.

1. (Optional) Test the ResellerSales column, such as to aggregate it by DimProduct[EnglishProductCategoryName], or refer to the "RELATEDTABLE" visual on the report page "Relating Data" in \Source\Part2\Adventure Works.pbix, which is shown in **Figure 5.3**.

EnglishProductCategoryName	ResellerSales
Accessories	$571,298
Bikes	$66,302,382
Clothing	$1,777,841
Components	$11,799,077
Total	**$80,450,597**

Figure 5.3 The ResellerSales calculated column uses the RELATEDTABLE function to aggregate related sales.

Analysis
The RELATEDTABLE function transitions the row context from DimProduct to FactResellerSales in order to filter the sales transactions for each product. Because it returns a table of matching rows, the SUMX extended function is used to iterate row by row over the returned table and summarize FactResellerSales[SalesAmount]. You can use COUNTROWS(RELATEDTABLE(FactResellerSales)) to see how many rows are matched.

Behind the scenes, RELATEDTABLE is a shortcut to the DAX CALCULATETABLE function. This formula produces the same result.

```
ResellerSales = SUMX(CALCULATETABLE(FactResellerSales), FactResellerSales[SalesAmount])
```

5.2 Navigating Virtual and Inactive Relationships

Active relationships are easy to work with and you should always create and use relationships when possible as they'll give you the best performance when joining tables. But what if two tables can't be related or they have an inactive relationship? Fortunately, DAX has functions to help you.

5.2.1 Looking up Values

You can use the LOOKUPVALUE function to look up a single value from another table. The LOOKUPVALUE function has the following definition:

```
LOOKUPVALUE( <result_columnName>, <search_columnName>, <search_value>[, <search_columnName>,
<search_value>]...[, <alternateResult>])
```

LOOKUPVALUE searches and returns the value in a column that meets one or more conditions. The search conditions must result in a single value or multiple (but identical) values to avoid an error. If no match is found, the function returns a blank value which you can substitute with another value specified in the alternateResult argument.

Practice
The DimEmployee table has a SalesTerritoryKey column that associates a sales person with a sales territory in the DimSalesTerritory table. Suppose you want to look up the assigned country and add it as a column to DimEmployee. Add a SalesTerritoryCountry calculated column to DimEmployee with this formula:

```
SalesTerritoryCountry = LOOKUPVALUE(DimSalesTerritory[SalesTerritoryCountry],
    DimSalesTerritory[SalesTerritoryKey], DimEmployee[SalesTerritoryKey])
```

Output
Most employees are not salespeople and they don't have an associated sales territory. In this case, the SalesTerritoryCountry calculated column shows 'NA' because that's the corresponding country for SalesTerritoryKey of 11. Otherwise, the column shows the associated country.

LastName	SalesTerritoryCountry
Anderson	NA
Ansman-Wolfe	United States
Arifin	NA
Bacon	NA
Baker	NA
Barbariol	NA

Figure 5.4 The SalesTerritoryCountry uses LOOKUPVALUE to look up the country associated with a salesperson.

Analysis
The first argument (DimSalesTerritory[SalesTerritoryCountry) is the value you want to retrieve. The second argument (DimSalesTerritory[SalesTerritoryKey]) is the column to search. The third argument is the search value. It must be a scalar expression that returns a single value whose type matches the column to be searched and cannot refer to any column in the searched table.

5.2.2 Navigating Inactive Relationships

Strictly speaking, the Adventure Works model has an inactive relationship between the DimEmployee and DimSalesTerritory tables. However, Power BI makes it difficult to use this relationship in calculated columns. For the sake of completeness, I'll provide the formula, but I recommend you use LOOKUPVALUE instead to avoid added complexity.

Practice
Unfortunately, RELATED and RELATEDTABLE can't navigate inactive relationships. The only way to navigate an inactive relationship is to use the USERELATIONSHIP function, which has the following definition:

```
USERELATIONSHIP(<columnName1>,<columnName2>)
```

The two columns must be from two different tables related with an inactive relationship. The first column should be on the many side of the relationship (foreign key), but Power BI will swap the columns if you change the order, so you don't have to remember this rule. The final formula is:

```
SalesTerritoryCountry = CALCULATE(
        CALCULATE(VALUES(DimSalesTerritory[SalesTerritoryCountry]), DimEmployee),
        USERELATIONSHIP(DimEmployee[SalesTerritoryKey], DimSalesTerritory[SalesTerritoryKey]))
```

Analysis
The outermost CALCULATE function transitions the row context to a filter context and uses USERELATIONSHIP as a second argument. The VALUES function returns the distinct values in a column, which in this case is the list of countries in the DimSalesTerritory[SalesTerritoryCountry] column. However, if you know that VALUES would return just one value, you can use it in a formula when a single value is expected.

So that only the associated country is returned, the formula uses a second CALCULATE function which filters only the countries (the outcome will be just one country) that intersects with the DimEmployee table over the inactive relationship. So, there are two context transitions:

1. From the row context of the current employee to the filter context over the inactive relationship.
2. A second context transition caused by the nested CALCULATE.

5.3 Summary

Calculated columns must often retrieve values from other tables. Use the RELATED and RELATEDTABLE functions to look up values from tables related with an active physical relationship. Use LOOKUPVALUE to lookup a single value when a physical relationship doesn't exist.