**Modeling Data in Power BI**

**By Stacia Misner Varga**

Contents

[1. Loading Data 2](#_Toc117087386)

[1.1 Supported Data Sources 2](#_Toc117087387)

[1.2 Counting Loaded Rows 3](#_Toc117087388)

[2. Exploring Additional Techniques in the Power Query Editor 4](#_Toc117087389)

[2.1 Commons Transformations 4](#_Toc117087390)

[2.2 Splitting Columns 4](#_Toc117087391)

[2.3 Understanding Custom Column 4](#_Toc117087392)

[2.4 Introducing the Star Schema 4](#_Toc117087393)

[2.5 Dimensions 4](#_Toc117087394)

[2.6 Facts 5](#_Toc117087395)

[3. Improving the Data Model for Reporting 6](#_Toc117087396)

[3.1 Configuring Model Properties 6](#_Toc117087397)

[3.2 Using Data Categories 6](#_Toc117087398)

[3.3 Understanding Q&A 6](#_Toc117087399)

[4. Using DAX to Enhance a Power BI Model 8](#_Toc117087400)

[4.1 Introducing DAX 8](#_Toc117087401)

[4.2 Using Variables in DAX Expressions 8](#_Toc117087402)

[4.3 Defining New Columns: Performing a Lookup to a Related Table 8](#_Toc117087403)

[4.4 Creating Measures 8](#_Toc117087404)

[4.5 Using DAX Parameters: Creating Query Parameters 9](#_Toc117087405)

[4.6 Using DAX Parameters: Working with What if Parameters 9](#_Toc117087406)

[5. Working with Dates and Time in Power BI 10](#_Toc117087407)

[5.1 Creating a Calculated Date Table 10](#_Toc117087408)

[5.2 Using DAX Functions for Date and Time Operations 10](#_Toc117087409)

[6. Applying Evaluation Context 11](#_Toc117087410)

[6.1 Introducing Row Context 11](#_Toc117087411)

[6.2 Working with Parent-child Functions 11](#_Toc117087412)

# 1. Loading Data

## 1.1 Supported Data Sources

* Databases: on-premises import (need to refresh this copy periodically) or direct connections.
* Files: files stored locally (computer’s network) or in OneDrive.
* Web: URL sources, OData feeds, and more.
* Azure: Cloud databases, storage, and services.
* Online Services: Software as a Service sources.
* Power Platform: Dataverse (formerly known as the Common Data Service), datasets, and dataflows.

For the best performance, data for Power BI is generally imported into your desktop application, but there’s a *data connectivity mode called DirectQuery*.

**DirectQuery:**

* This option only works with some databases, and you’ll know which ones they are when you see that data connectivity mode option when you’re importing data.
* With DirectQuery, the data does not move out of the source database. Instead, Power BI queries the database when you view the report. So the data is always current.
* In some cases, Power BI gateway may be required.
* Limited calculation column support.
* DirectQuery can be a great option when your dataset is really large or when you need near real-time data access.

*You do have a different kind of data connectivity option for other data sources called live connection (connect live).*

**Connect Live:**

* It’s like DirectQuery because the model stores only the connection and the data stays in the source, it does not get copied over into Power BI.
* Using this connection, you’ll be able to connect to tabular models (Analysis Services Tabular), as wells as multidimensional models (Analysis Services Multidimensional).
* You can work with the data directly in Power BI without making any changes to the model itself inside of Power BI.
* You do need to have Power BI gateway if you plan to publish reports to the Power BI service so that Power BI in the cloud can reach into your on-premises environment.

**Composite Model:**

* Before, your model could only contain imported data OR a connection to a DirectQuery OR Connect Live source. You could not combine the two. But, composite models were introduced in Power BI in 2018 to support a mix of these environments.
* Use DirectQuery/Connect Live for large data sets.
* Use an imported source for smaller data sets used to group or filter the large data sets.
* Implement aggregation tables to improve performance.

## 1.2 Counting Loaded Rows

**Measure:** is a numeric value that represents a calculation based on the data in your model.

# 2. Exploring Additional Techniques in the Power Query Editor

## 2.1 Commons Transformations

* Rename your queries.
* Eliminating columns.
* Filtering rows.
* Renaming columns.
* Fixing data types.
* Replacing values.

## 2.2 Splitting Columns

Split Column:

* By delimiter
* By number of characters
* By position
* By case – upper to lower or lower to upper
* By digit to non-digit or non-digit to digit

## 2.3 Understanding Custom Column

Custom Column:

* Column from example
* Custom formula
* Custom function invocation
* Conditional column
* Index column
* Duplicate column

## Introducing the Star Schema

Star schema: is a collection of dimension and fact tables. Star schema is a term used to define a collection of tables and how they relate to one another. In the simplest terms, a Star Schema contains two kinds of tables – dimensions and facts.

* Dimensions contain data about people, places, and things.
* Facts contain the numerical data that we use to quantify dimension data such as revenue and profit.
* Star Schema design is commonly used in data warehouse projects for relational databases. It’s a mature and effective system for organizing data structures specifically for reporting and analytical applications.
* Optimal design for Power BI performance and usability. It helps Power BI perform better, and it organizes your data in a more usable way for reporting.
* Also known as dimensional modeling.

## 2.5 Dimensions

* Dimensions are what we call the nouns of the model. At minimum you use columns in a dimension to describe these nouns.
* Columns in a dimension can serve a purpose besides description. You can use them to group.
* Dimension columns can also be used for filtering.

## 2.6 Facts

Kinds of design:

* Regular fact table: its job is to hold the numbers.
* Factless fact table: sometimes they’re called bridge tables and what makes them factless is that they don’t have values in them. They really serve to bridge together two dimensions.

# 3. Improving the Data Model for Reporting

## 3.1 Configuring Model Properties

Steps to review the model properties:

* **Object names and descriptions:** first you can configure metadata for model objects, specifically tables or fields in a table. This could be as simple as renaming these objects, and you can also add descriptions to your fields to make it easier for people to understand what a particular field represents. Make sure that the names of tables makes sente to yourself and to the users.
* **Hidden objects:** sometimes tables or fields are in your model because they’re needed to set up relationships or maybe to compute column values, but they’re not necessarily useful as individual objects that you need to add to a visualization so you can hide them.
* **Sort order:** sometimes you might need to adjust the sort order, especially when you have columns that you don’t want to sort alphabetically, such as months in the year or sizes.
* **Data types:** fix data types individually over, and over, and over.
* **Format:** for numeric columns, you should fix the formatting (currency, commas, decimal places, etc.)
* **Default summarization:** Power BI wants to sum up numeric values and it sets that behavior by default, but sometimes that’s not the way you want to summarize values. Even with numbers in the column, maybe it’s not appropriate to summarize those values at all, such as when you have an identifier number or maybe a year.

## 3.2 Using Data Categories

Power BI uses information in the data model to decide how to display data by default, such as aggregating numeric values and then displaying the results in a table or a chart. In other specific cases, you can add some additional control over data behavior by assigning a field to a data category, and there are three types:

* **Geographic location:** by assigning a field to a type of geographical location, such as address or continent, you can eliminate some ambiguity. And you also have the ability to use precise geographical data such as latitude and longitude.
* **URL**: there are URLs for web pages and URLs for images. The key to using these URLs in Power BI is that you have to be able to access that URL from your computer. And if you publish your file to the Power BI service in the cloud, those URLs also need to be accessible publicly.
* **Barcode**: you need to put data into a column that relates to a barcode, and then Power BI uses this data as a filter. But this only works when you have the Power BI mobile app on a mobile device, and then you’d literally use the camera on your mobile device to scan a barcode on a product or maybe on a document, and then you can display your Power BI report for that scanned item based on its barcode. In other words, you’re scanning to get a filter set in your report.

## 3.3 Understanding Q&A

* Q&A is available both in the Power BI service, as well as in the Power BI Desktop application.
* Its purpose is to support **natural language queries** such as which movie is longest, for example.
* For this feature to work properly, the model has to meet certain requirements:
  + Correct relationships and properties.
  + It’s important to normalize the **model structure** as much as possible, meaning you should have a star schema. Power BI behaves much better with a star schema structure when it comes to using Q&A.
* **Enhancement options**: sometimes Power BI is not able to guess correctly what a question is asking for, and you have to help it along.
  + One way to do that is to add **synonyms**, which essentially expands the vocabulary about the model that Power BI understands.
  + And if you really want to get in depth with enhancements, you can edit the **linguistic schema** that Power BI uses to translate your questions into targeted objects and structuring the queries.

# 4. Using DAX to Enhance a Power BI Model

## 4.1 Introducing DAX

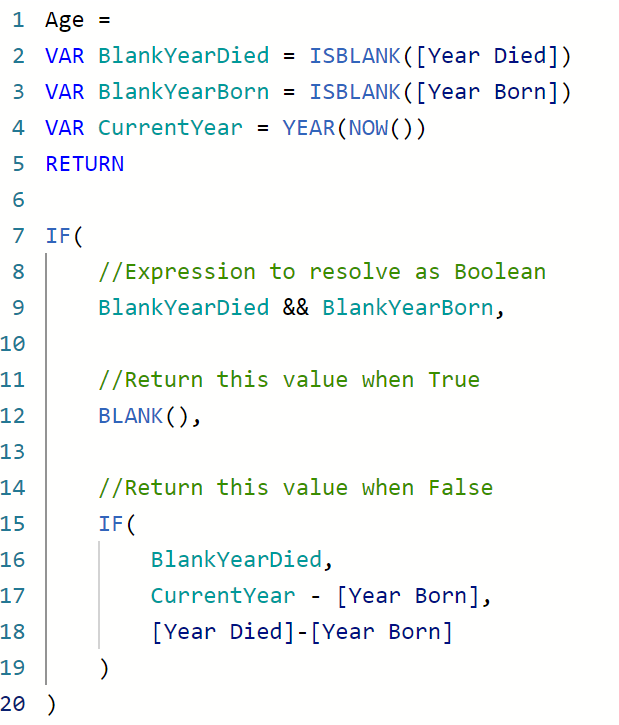
**DAX:** Data Analysis Expressions is a language for defining columns and measures in Power BI data models.

Comparing Excel and DAX Functions:

* Excel functions operate on individual cells. Cells contain values and you can reference those values by using a variable for the cell which represents its coordinate, and you can reference those cells and formulas that use operators like addition and multiplication, or you can reference them in functions. And you can also operate on ranges of cells where you specify the beginning and endpoints of the range.
* DAX functions, on the other hand, operate exclusively on tables or columns and that’s it. So, for example the COUNTROWS functions looks at the entire table (“=COUNTROWS(‘Movie Values’)”) or DAX can operate on columns. The big learning curve with DAX is learning to think about applying operations to just tables or just columns, there is no access to individual cells.

## 4.2 Using Variables in DAX Expressions

You can use Variables in DAX Expressions, follow the example:



You don’t have to specify any type of data type for the variable.

## 4.3 Defining New Columns: Performing a Lookup to a Related Table

**RELATED function:** its purpose is to return a related value from another table. The RELATED function requires a columns name as an argument, but you need to tell it first which table contains the column.

Ex: RELATED(‘Movie Values’[Release Date])

## 4.4 Creating Measures

Measures vs. Calculated Columns:

**Measures:**

* Calculated on addition to visualization
* Table name not required in expression
* Name must be unique within model
* No impact on memory

**Calculated Columns:**

* Calculated on addition to model
* Table name might be required
* Name must be unique within table
* Increases memory used by model

## 4.5 Using DAX Parameters: Creating Query Parameters

**Query Parameter:** pass a user selection to a query and reference the selection in a DAX expression.

## 4.6 Using DAX Parameters: Working with What if Parameters

**What if Parameter:** reference a user selection in DAX expression to dynamically update a visualization.

# 5. Working with Dates and Time in Power BI

## 5.1 Creating a Calculated Date Table

**CALENDAR() function:** returns a table containing all dates between a specific start and end date that you hard code into the expression.

**CALENDARAUTO() function:** that looks at the CALENDAR() function and the last dates found anywhere in the model based on the date data type. And then it generates a table of all the dates between that first and last date.

## 5.2 Using DAX Functions for Date and Time Operations

Functions for Date and Time:

* **= DATE([Year], [Month Number], [Day Number of Month])** -> Numeric to datetime data type conversion by year, month, and day.
* **=DATEVALUE(“1/1/2011”)** -> Text to datetime data type conversion by string value “DD/MM/YYY”.
* **=TIME([Hour], [Minute], [Second])** -> Numeric to datetime data type conversion by hour, minute, and second.
* **TIMEVALUE(“12:00:00”)** -> Text to datetime data type conversion by string value “HH:MM:SS”.

Datepart extraction from datetime:

* =DAY([Date])
* =MONTH([Date])
* =YEAR([Date])
* =WEEKDAY([Date], 2)
* =WEEKNUM([Date], 2)

Timepart extraction from datetime:

* =HOUR([StartTime])
* =MINUTE([StartTime])
* =SECOND([StartTime])

DAX functions for date:

* **=TODAY()** or **=UTCTODAY()** -> Current date
* **=NOW()** or **=UTCNOW()** -> Current time
* **=EDATE([StartDate], 3)** or **=EOMONTH([StartDate], 3)** -> Relative date calculations
* **=YEARFRAC([StartDate], [EndDate])** -> Partial year calculation
* **=DATEDIFF([StartDate], [EndDate], YEAR)** -> Date interval (SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, and YEAR)

# 6. Applying Evaluation Context

## 6.1 Introducing Row Context

**Row context:** It defines which columns are related to an active row and in scope for a given calculation.

**Calculated columns:** when you add an expression to a calculated column, you can see a result populate in that column for every single row, even an empty result or an error is still a result.

**DAX iterator functions:** perform row context calculations then aggregate results.

## 6.2 Working with Parent-child Functions

* Use for structures with varying levels like organization charts and charts of accounts.
* Create a calculated column for each level of the parent-child hierarchy.
* Use DAX parent-child functions to traverse the lineage to the specified level.
* Add hidden measures to hide blank values for levels without members.