Module 4

Working with Power BI Models and DAX

Module Overview

- The Power BI Data Model
- Relationships
- DAX queries
- Calculations and measures

Lesson 1: The Power BI data model

- What is a data model?
- Managing data relationships
- Optimizing the model for reporting
- What are hierarchies?
- Creating hierarchies

What is a data model?

- Data model typically associated with relational database
- In Power BI, connect to multiple different data sources and import into data model
- Shape optimized data ready for using in reports
- No need to flatten data
  - **Data types**: ensure data uses correct data type
  - **Fact and dimension tables**: create star schema in model
  - **Cross filtering**: bi-directional to flatten tables
  - **Reduce size of dataset**: exclude columns or rows for very large databases, or omit sensitive data

Managing data relationships

- Relationships are either created automatically by Power BI, or you create them manually
  - **Primary keys**: uniquely identify each row in a table; surrogate key based on nonbusiness data, such as an incrementing whole number
  - **Foreign keys**: join to the primary key table to create relationship; ensure data integrity and prevent deletions in the primary key table
  - **Creating relationships**: Power BI creates any apparent relationships when data is imported
  - **Viewing relationships**: view and manage relationships in the data mode in a diagrammatic view

Optimizing the model for reporting

- Imported data from a database is often very raw:
  - Data in the model might not be formatted or optimized
  - Inconsistency in data types for data from different sources
- Optimize data in model:
  - **Hide fields**: hide fields not used in visuals in report; makes model easier to use, useful for large tables
  - **Sort data**: display data in correct order in visuals, such as ordering by day name or month name
  - **Format data**: change data types and formatting; especially useful for datetime and currency fields

What are hierarchies?

- Hierarchies enable drill-down into your data
- A hierarchy is a set of related fields grouped together
- Each level is contained within the next level and does not exist independently
  - Example: Country, State, City
- Time intelligence: Power BI automatically creates drill-down on date columns for:
  - Year, Quarter, Month, Day
- Can also use DAX time intelligence functions to aggregate date for specific time periods

Creating hierarchies

- Create hierarchies in Power BI **FIELDS** pane:
  - Select column and click **New hierarchy**
  - Right-click and click **Rename** to specify new name
  - Click ellipsis of another column, and **Add to hierarchy**
  - Repeat to add columns; can also move, delete, rename
- Drag hierarchy to **Axis** field bucket of a visual
  - Creates navigable hierarchy in the visual
  - Click the button **Click to turn on Drill Down**
  - Click data points to drill down into data
  - Expand to see all data in a level
  - Use filters in any level to exclude data from visual

Demonstration and Exercise 1

You will see how to:

- Create a hierarchy
- Use the hierarchy to navigate data

Lesson 2: Relationships

- What are relationships?
- Viewing relationships
- Creating relationships
- Cardinality
- Cross filter direction

What are relationships?

- Relationships join tables together so you can work with multiple tables as if they were one:
  - Usually created in an OLTP system as part of the normalization process, by adding keys to tables
  - Prevents repeated values, and each entity has only those attributes that belong to it
  - Data warehouse uses fact tables, with keys that join to dimension tables
  - Power BI Autodetect feature can recognize relationships, and creates them automatically

Viewing relationships

- Power BI Autodetect feature works out relationships in queries run against data source:
  - Relationships are created automatically after data load
  - Autodetect determines cardinality and cross filter direction in the relationship
  - View and edit relationships created by Power BI in the Model view, using a relationship diagram
  - When Power BI detects more than one relationship between two tables, only one can be active, and is set as the default; turn off incorrect active relationship
  - Delete relationships in the Model view

Creating relationships

- Two ways to create relationships in Power BI:
  - Using Power BI Autodetect feature
  - Create relationship manually

- If you can't create a relationship, it's likely to be because of null, or empty values, or duplicate rows

Cardinality

- Cardinality is the relationship between two tables

- Power BI supports three types:
  - Many to one (*:1):
  - One to one (1:1):
  - One to many (1:*):

Cross filter direction

- The cross filter direction of relationships affects how Power BI treats the tables in visualizations:
  - The cross filter direction is automatically set when relationships are created manually or using Autodetect
  - Power BI makes best guess at the direction

- Two types of cross filter direction:
  - Both
  - Single

Demonstration and Exercise 2

You will see how to:

- Import a data extract into Power BI
- View and edit the relationships created automatically
- Add new relationships
- See how Cross Filters work

Lesson 3: DAX queries

- Overview of DAX
- What is DAX?
- Syntax
- Functions
- Context

Overview of DAX

- Formula-based language for building business logic and queries in tabular data models:
  - Create columns and measures based on complex custom calculations
  - Create columns that reference data in related tables
- Syntax is similar to Microsoft Excel formulas
- Not a new feature—used in other products
  - Power Pivot for Excel
  - SQL Server Analysis Services (SSAS) Tabular
  - Power BI
- Designed to work with relational data

What is DAX?

- Data Analysis Expressions (DAX) is a formula language:
  - Comprises a library of more than 200 functions, constants, and operators
  - DAX Queries:
    - Calculates returns multiple values in table format
  - DAX Expressions
    - Calculates and returns a single value
    - Can Create:
      - Calculated Columns
      - Calculated Tables
      - Measure

DAX Queries

- Client applications, such as the Power View reporting tool, issue DAX queries
- You can write queries manually by using the DAX query language
- You can filter, order, and summarize results
- Begins with EVALUATE keyword

DAX Queries Examples

```
EVALUATE CALCULATETABLE('Product', 'Product'[Color] = "Blue")
        ORDER BY 'Product'[Product Name],
                 'Product'[List Price]
```

```
DEFINE
        VAR MinAmt = 10000
        VAR MaxAmt = 10000000
EVALUATE FILTER(
                ADDCOLUMNS(
                  SUMMARIZE('Sales', 'Product'[Color]),
                    "Total Amount", [TotalAmt]),
                AND ( [Total Amount] >= MinAmt,
                      [Total Amount] <= MaxAmt)
                )
```

DAX Expression Syntax

- DAX formulas must be syntactically correct before you can save them to the model:
  - The first part of the formula is the name of the measure, or calculated column
    - This is followed by the equal operator (=)
    - The equal operator returns the result of the calculation to its right, back to the measure (much like a variable)
    - Functions must have at least one argument passed to it in parentheses ()
  - Include table and column name:
    - Column name must be enclosed in square brackets []
    - Table names with spaces or reserved words must be enclosed with single quotation marks (')
  - Measures created in context of current table—can move
    - Best Practice: Always include table reference

Functions

- DAX functions are predefined formulas that perform calculations on one or more arguments:
  - You pass a column, function, expression, formula, constant, number, text, TRUE or FALSE as arguments
  - DAX functions are similar to Excel except:
    - They reference an entire column or table;
    - Use filters to reference selected values
  - Functions that return a table do not display results
  - VLOOKUP is effectively replaced with relational data model

DAX Functions

- Text functions
- Information functions
- Filter and value functions
- Logical functions
- Mathematical and trigonometric functions
- Statistical and aggregation functions
- Date and time functions
- Time intelligence functions
- Navigation and Lookup Functions

DAX Expression Examples

- Simple Measure

```
Total Sales = SUM('InternetSales'[LineTotal])
```

- Calculated Column

```
Full Name = [First Name] & " " & [Last Name]
Location = RELATED('Countries'[Region]) & ", " & [City]
```

- Filtering

```
Europe Sales = CALCULATE([Total Sales], 'Territory'[Group] = "Europe")
```

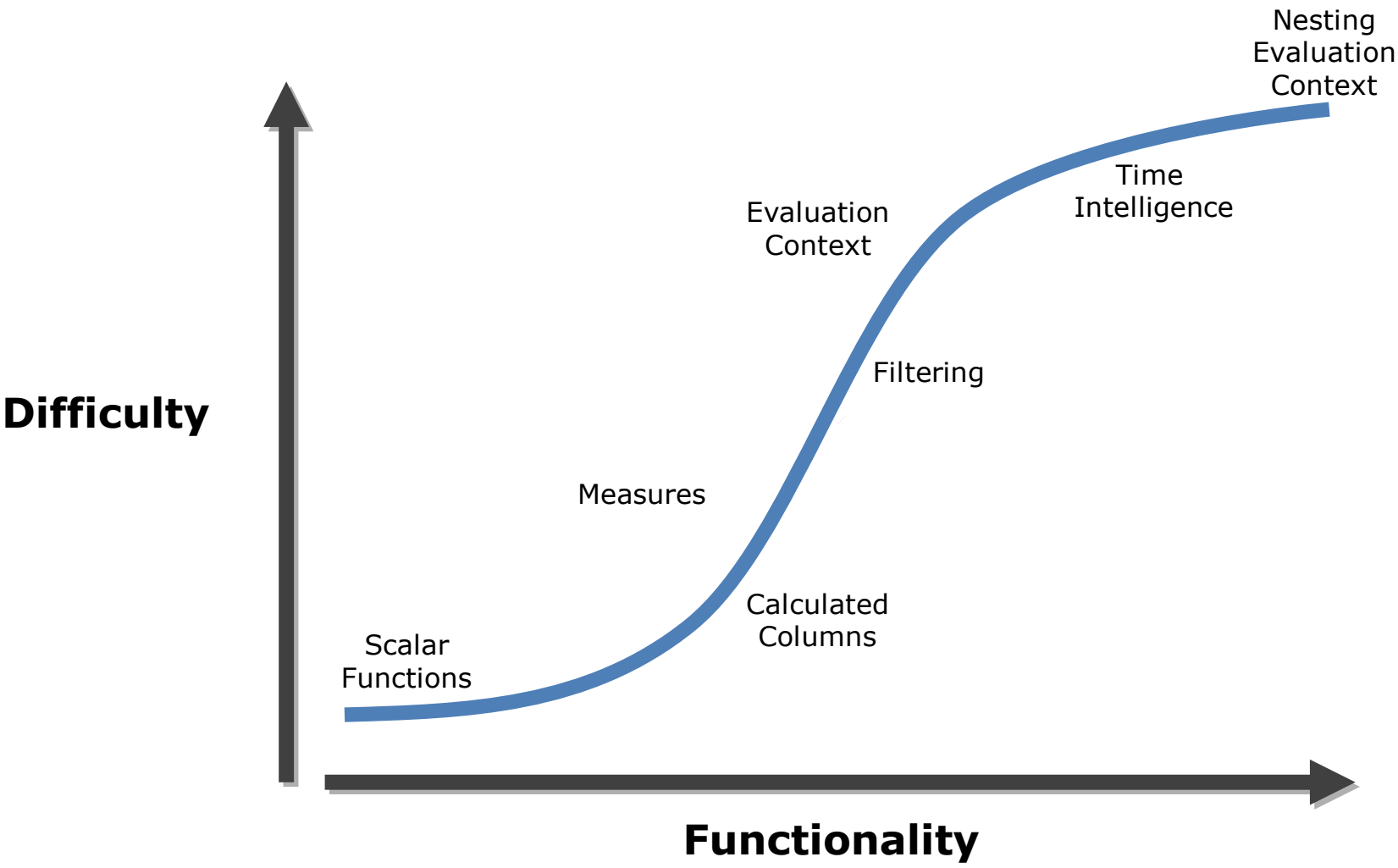
- Time Intelligence

```
Last Year Sales = CALCULATE([Total Sales],
SAMEPERIODLASTYEAR('Date'[Date]))
```

DAX Data Types, Operators, and Functions

- DAX Data Types
  - https://docs.microsoft.com/en-us/power-bi/desktop-data-types
- DAX Operators
  - https://docs.microsoft.com/en-us/dax/dax-operator-reference
- DAX Functions
  - https://docs.microsoft.com/en-us/dax/dax-syntax-reference

DAX Learning Curve



Eugene Meidinger
Pluralsight

Evaluation Context

- DAX expressions use two types of context:
  - Row context:
    - Row context is the current row
    - Often applied to measures to identify a single row
  - Filter context:
    - Exists in addition to row context
    - A filter context is one or more filters applied in a calculation to determine the single value or result
    - Filter contexts are used in visualizations; for example:
      - A chart with Sales, Sales Person, and Month.
      - The chart returns subsets of data based on a specific Sales Person, and Month
    - You can apply filter contexts using visualizations, and DAX

Row vs Filter Context

- The filter context filters data, whereas the row context iterates tables.
- When DAX is iterating, it is not filtering; and when it is filtering, it is not iterating.
- The filter context filters, the row context iterates, they are not the same.

  - Marco Russo - The Definitive Guide to DAX (p. 80)

Lesson 4: Calculations and measures

- Calculated columns
- Measures
- Calculated tables

Calculated columns

- Calculated columns are added to tables using DAX formulas to perform operations on existing data:
  - New column using data inside the model
  - Concatenate strings, calculate numbers, or combine data from elsewhere in the model
  - Used in Visualizations on an Axis, Legend or Group
  - Use New Column on Data or Report view to create column
  - Use in Visualizations like any other column

```
Full Name = [First Name] & " " & [Last Name]
Location = RELATED('Countries'[Region]) & ", " & [City]
```

Measures

- Measures help you discover insights into your data that might otherwise be hidden:
  - Use aggregations: SUM, AVG, MIN, MAX, DISTINCTCOUNT
  - Can create complex calculations
  - Generally used in Values area of Visualizations
  - Can create in Data or Report view
  - Can be assigned to any table the Home table

```
Total Sales = SUM(Sales[SalesAmount])
Europe Sales = CALCULATE([Total Sales],
        Territory[Group] = "Europe")
Due Date Sales = CALCULATE( SUM([Total Sales]),
        USERRELATIONSHIP( Sales[DueDate], Date[Date]))
```

Calculated tables

- Create calculated tables using data that exists in the model:
  - Create table in Reporting or Data view
    - From the **Modeling** tab, in the **Calculations** group, click **New Table**, and then add DAX formula
  - Use data from the model to create the new table, rather than querying the data source
  - Combine tables using functions
    - UNION, NATURALINNERJOIN, and NATURALLEFTOUTERJOIN
    - Currently doesn't work
  - Calculated tables and columns are used in the same way as other tables. Rename table and columns, use in relationships with other tables, change data types, add columns, measures, and use in visualizations

Calculated Tables Examples

- From Existing Table

```
Due Date = SELECTCOLUMNS(Date, "Date", Date[FullDate], "Year",
Date[Calendar Year], "Qtr", Date[Calendar Qtr])
```

- Static Table

```
Lookup = DATATABLE("Key", Integer, "Name" String,
        { {1, "Joe"}, {2, "Mary"}, {3, "Bill"}, {4, "Jane"}})
```

- Generated Table

```
Sequence = GENERATESERIES(1, 1000, 1)
```

Demonstration and Exercise 3

You will see how to:

- Create calculated columns
- Make calculated columns include columns from other tables
- Create hierarchies across tables
- Create measures with simple scalars
- Create measures based on other measures
- Utilize Time Intelligence
- Activate relationships with a measure