e-Book

DAX mit Power BI

Webshop Ausgabe

Grundlagen und DAX Patterns für zahlreiche praktische Aufgabenstellungen

Ausgabe vom 11.06.2021

Erstellt von Mag. Robert Lochner

www.linearis.at

#linearisblog

Power BI Camp

RECHTLICHES

© Robert Lochner, Linearis GmbH, 1020 Wien

Dieses e-Book darf ausschließlich durch den in der Fußzeile angegebenen Lizenzerwerber genutzt werden.

Das Werk einschließlich aller Inhalte ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Nachdruck, Reproduktion (auch auszugsweise) in irgendeiner Form (Druck, Fotokopie oder anderes Verfahren) sowie die Einspeicherung, Verarbeitung, Vervielfältigung und Verbreitung mit Hilfe elektronischer Systeme jeglicher Art, gesamt oder auszugsweise, ist ohne ausdrückliche schriftliche Genehmigung des Autors untersagt. Alle Übersetzungsrechte vorbehalten.

Die Benutzung dieses e-Books und die Umsetzung der darin enthaltenen Informationen erfolgt ausdrücklich auf eigenes Risiko. Das Werk inklusive aller Inhalte wurde unter größter Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden. Der Autor übernimmt keine Haftung für die Aktualität, Richtigkeit und Vollständigkeit der Inhalte des e-Books. Es kann keine juristische Verantwortung sowie Haftung in irgendeiner Form für fehlerhafte Angaben und daraus entstandenen Folgen vom Autor übernommen werden. Für die Inhalte von den in diesem Buch abgedruckten Internetseiten sind ausschließlich die Betreiber der jeweiligen Internetseiten verantwortlich.

Power BI, Excel Power Pivot, SQL Server Tabular Model und Microsoft sind eingetragene Warenzeichen der Microsoft Corporation, Redmond.

POWER BI CAMP

Dieses e-Book ist als Manual für das Trainingsmodul „DAX & Datenmodellierung mit Power BI" im Rahmen des „Power BI Camp" konzipiert.

Bitte beachten Sie, daß nur die Teilnehmer eines Trainings den vollen Umfang der DAX Patterns in diesem e-Book freigeschalten bekommen und nur diese Teilnehmer die Trainingscases im PBIX-Format erhalten.

Rechtliches ***2***
Power BI Camp ***2***
1 Einleitung ***6***
1.1 Ziele dieses e-Books **6**
1.2 Nicht-Ziele **6**
1.3 Sprachliche Konventionen **6**
1.4 Power BI Desktop Version **7**
1.5 Verwendete Shortcuts **7**
2 Power BI Grundlagen ***8***
2.1 Was ist Power BI? **8**
2.2 Power BI im Kontext des Microsoft Produktportfolios **9**
2.3 Power BI Architektur und Fokus dieses e-Books **10**
2.4 User Interface von Power BI Desktop **11**
2.5 Zentrale Begriffe im Datamodel **12**
3 DAX Grundlagen ***13***
3.1 Einstieg in die DAX Formelsprache **13**
3.1.1 Aufgabenstellung 13
3.1.2 Lösung in Excel – SUMME(), SVERWEIS() und SUMMEWENN() 14
3.1.3 Lösung in Power BI – Erarbeitung der DAX Fundamentals 15
3.1.4 Charakteristiken der DAX Formelsprache 22
3.2 DAX Instrumente ("Baustein 1") **23**
3.2.1 Instrument "Measures" 24
3.2.2 Instrument "Calculated Columns" 25
3.2.3 Instrument "Calculated Tables" 26
3.2.4 Instrument "Row Level Security (RLS)" 27
3.2.5 Instrument "DAX Queries" 28
3.2.6 Abgrenzungen 29
3.3 DAX Evaluation Context („Baustein 2") 30
3.3.1 DAX Formula Evaluation 30
3.3.2 Konzept des Filter Context 31
3.3.3 Konzept des Row Context 32
3.3.4 Evaluation Matrix – Stufe 1 und 2 33
3.3.5 Konzept der Context Transition 34
3.3.6 Konzept des Outer Filter Context 35
3.3.7 Konzept der Expanded Tables 36
3.3.8 Evaluation Matrix – Stufe 3 38
3.4 DAX Syntax ("Baustein 3") **39**
3.4.1 Übersicht 39
3.4.2 Allgemeine DAX Syntax Regeln 39
3.4.3 Datentypen in DAX 42
3.4.4 Variablen in DAX 44
3.4.5 Skalare in DAX 46
3.4.6 Syntax Sugar 47
3.5 Zentrale DAX Funktionen **50**
3.5.1 Aggregatorfunktionen 50
3.5.2 Iteratorfunktionen 51
3.5.3 CALCULATE() 52
3.5.4 Tabellenfunktionen 55

3.5.5 Time Intelligence Funktionen 57
3.5.6 Logikfunktionen: IF() vs. SWITCH() 59
3.5.7 Statusfunktionen: BLANK() 60
3.5.8 Informationsfunktionen 60
3.5.9 Lookupfunktionen 61

4 DAX Patterns – Measures 66

4.1 M01 Aggregationen und Blanks | SUM(), AVERAGE(), MIN(), MAX() und BLANK() 67

4.2 M02 Zählen und Blanks | COUNTROWS(), COUNT(), BLANK(), ISBLANK() 69

4.3 M03 Gewichteter Durchschnittspreis (Measure vs. Column) | DIVIDE() 70

4.4 M04 Anzahl Kunden, Länder, usw. | DISTINCTCOUNT() 71

4.4.1 M04.1 Anzahl Kunden, Tage, Artikel, usw. = Ermittlung in der Faktentabelle | DISTINCTCOUNT() 71
4.4.2 M04.2 Anzahl Länder, Produktgruppen, Monate, usw. = Ermittlung in der Dimensionstabelle | DISTINCTCOUNT(), CALCULATE(), CROSSFILTER() 72

4.5 M05 Gefilterte Werte | CALCULATE(), SUMX(), ALL(), KEEPFILTERS() und FILTER() 74

4.5.1 M05.1 Gefilterte Werte Allgemein 75
4.5.2 M05.2 Gefilterte Werte IST-PLAN 79

4.6 M06 %-von-Gesamt Berechnungen | CALCULATE() 80

4.6.1 M06.1 %-von-Gesamt absolut | CALCULATE(), ALL(), REMOVEFILTERS() 81
4.6.2 M06.2 %- von-Gesamt relativ | CALCULATE(), ALL(Column) und ALLSELECTED(Column) 82
4.6.3 M06.3 %- von-Gesamt jahresweise | CALCULATE und ALLEXCEPT(Column) 84
4.6.4 M06.4 %- von-Umsatz Berechnung | CALCULATE(), FILTER(), ALLSELECTED() 85

4.7 M07 Time Intelligence Berechnungen | CALCULATE() 86

4.7.1 M07.1 Year-to-Date | TOTALYTD(), DATESYTD() 86
4.7.2 M07.2 Year-over-Year | SAMEPERIODLASTYEAR(), DATEADD() 88
4.7.3 M07.3 Period-over-Period | DATEADD(), PREVIOUSx(), PARALLELPERIOD() 89
4.7.4 M07.4 Week-over-Week mit Max-/Ebenen-Limitierung | SUMX(), FILTER(), LOOKUPVALUE(), HASONEVALUE() 90
4.7.5 M07.5 Change over Previous-N-Days mit Ebenen-Limitierung | DATESBETWEEEN(), HASONEVALUE() 93
4.7.6 M07.6 Moving-Annual-Total (MAT) mit Min-/Max-Limitierung | DATESINPERIOD() 94
4.7.7 M07.7 Moving-12-Months-Average | DATESINPERIOD() Expanded Tables 95
4.7.8 M07.8 Month-to-Date Berechnung in Detaillisten | CALCULATE(), DATESMTD(), ALL(), ALLSELECTED(), DATESBETWEEN(), FILTER() 96

4.8 M08 Bestandslogik (Semi-Additive Measures) | CALCULATE() 101

4.8.1 M08.1 Bestandslogik für Bestandsgrößen | LASTDATE(), LASTNONBLANK(), DATESBETWEEN() 101
4.8.2 M08.2 Bestandslogik für Flußgrößen | CALCULATE() 105

4.9 M09 Iteratoren 106

4.9.1 M09.1 Menge x Preis zur Umsatzermittlung | SUMX(), RELATED() 106
4.9.2 M09.2 Größte/kleinste/durchschnittliche Tagesumsätze pro Monat | MINX(), MAXX(), AVERAGEX() 107
4.9.3 M09.3 Korrekte Ermittlung auf Total-Ebene | SUMX() 108
4.9.4 M09.4 Korrekte Aggregation vervielfachter Werter | AVERAGE(), SUMX() 109
4.9.5 M09.5 Anzeige der selektierten Werte als Textkette | CONCATENATEX(), VALUES(), CROSSFILTER(), SUMMARIZE() 111

4.10 Weitere Patterns 113

5 DAX Patterns – Calculated Columns 113

5.1 C01 Preisberechnung auf Datensatzebene (Erlös/Menge) | DIVIDE() 114

5.2 C02 Attribute in Datumsdimension hinzufügen | Datumsfunktionen YEAR(), QUARTER(), MONTH(), WEEKNUM(), WEEKDAY() und FORMAT() 114

5.3 C03 Textfelder zerlegen oder verketten | Textfunktionen LEFT(), RIGHT(), MID(), FIND() und CONCATENATE() 115

5.4 C04 Statische Altersberechnung 116

5.4.1 C04.1 Altersberechnung in Dimensionstabelle | IF(), DATEDIFF(), YEARFRAC(), ROUNDDOWN(), AVERAGE(), CROSSFILTER() 116
5.4.2 C04.2 Altersberechnung in Faktentabelle | IF(), AVERAGE() 119
5.5 C05 Auflösen von Parent-Child Strukturen | PATH(), PATHITEM(), LOOKUPVALUE() 121
5.6 Weitere Patterns 123
6 DAX Patterns – Calculated Tables 123
6.1 T01 Hard-codierte Tabelle | Table Constructor und DATATABLE() 124
6.2 T02 Berechnete Parameter Tabelle | GENERATESERIES() 126
6.3 T03 Berechnete Datumsdimension 128
6.3.1 T03.1 Erstellen der Datumsspalte | CALENDAR() und CALENDARAUTO() 128
6.3.2 T03.2 Attribute für die Datumsdimension | ADDCOLUMNS() 130
6.4 T04 Duplizieren einer bestehenden Tabelle | ALL() und ALLNOBLANKROW() 132
6.5 Weitere Patterns 133
7 DAX Patterns – Row Level Security 133
7.0 Allgemeines zur Row Level Security 134
7.1 R01 Statische RLS 138
7.1.1 R01.1 Statische RLS mit UND-Bedingungen | (Filterbedingungen) 138
7.1.2 R01.2 Statische RLS mit ODER-Bedingungen | (Filterbedingungen) 140
7.2 R02 Dynamische RLS mit 1:n Beziehung 141
7.2.1 R02.1 Dynamische RLS mit 1:n und Top Management Role | USERPRINCIPALNAME() 141
7.2.2 R02.2 Dynamische RLS mit 1:n und Top Management Flag | USERPRINCIPALNAME(), MAXX(), FILTER() 142
7.3 R03 Dynamische RLS mit m:n Pattern 143
7.3.1 R03.1 Dynamische RLS mit m:n bidirektionaler Beziehung | USERPRINCIPALNAME() 143
7.3.2 R03.2 Dynamische RLS mit m:n Filterbedingung | USERPRINCIPALNAME(), IN(), SELECTCOLUMNS(), FILTER() 145
7.3.3 R03.3 Dynamische RLS mit m:n Lookup Column | USERPRINCIPALNAME(), NOT(), ISBLANK(), LOOKUPVALUE() 147
7.4 Weitere Patterns 149
8 Informationen zu den Trainingscases 149
8.1 Trainingscase „AW Reseller Sales" 149
8.2 Trainingscase „Headcount Reporting" 151
8.3 Trainingscase „HR Reporting" 152
8.4 Trainingscase „Headcount aus Zeiträumen" 152
8.5 Trainingscase „P&L Reporting" 153
8.6 Trainingscase „Sales Report" 154
8.7 Trainingscase „Vervielfachte Werte" 154
8.8 Trainingscase „Multi-Fakten Schema Typ 1" 155
8.9 Trainingscase „Semi-additive Kennzahlen" 155
8.10 Trainingscase „Simple Tabelle" 155
8.11 Trainingscase „Survey Data" 156
9 Cheat Sheet & Best Practices 157
10 Quellen 158
10.1 DAX Referenzen 158
10.2 Bücher und Blogs 159

1 EINLEITUNG

1.1 ZIELE DIESES E-BOOKS

Ziel dieses E-Books ist es, eine solide Basis zur Verwendung der **DAX Engine von Power BI Desktop** zu liefern sowie Ideen, Anregungen und weiterführende Links zu liefern:

a. Solides Verständnis zu den **DAX Grundlagen** (Instrumente, Syntax, Evaluation Context)

b. Lösungsbausteine für zahlreiche Praxisfälle
   - **DAX Patterns** für Measures
   - **DAX Patterns** für Calculated Columns
   - **DAX Patterns** für Calculated Tables
   - **DAX Patterns** für Row Level Security

c. Zusätzliches Know-How zu **ausgewählten DAX-Funktionen** (wobei das praxisrelevante Know-How bereits in den Patterns „verpackt" ist)

1.2 NICHT-ZIELE

Der Fokus liegt dabei nicht darauf, eine detaillierte Beschreibung der konkreten Umsetzung im Programm Power BI Desktop zu liefern (hier verweisen wir auf die Programmdokumentation) und auch nicht eine detaillierte Beschreibung von DAX-Funktionen (hier verweisen wir auf die DAX-Funktionsdokumentation). Stattdessen werden Grundlagen und Patterns für die Praxis geliefert, deren Verständnis dann im Zuge weiterer eigener Recherchen vertieft werden können.

Ebenso wird keine durchgehende Story („Trainingsfahrplan") erzählt, sondern dieses e-Book ist bewußt modular nach DAX Patterns aufgebaut und eignet sich daher sehr gut als Nachschlagewerk für die Praxis und nur bedingt als Trainingshandbuch für Einsteiger. Dies ist auch wichtig, um das e-Book in regelmäßigen Abständen an die laufend erscheinenden neuen Releases von Power BI Desktop anpassen zu können.

1.3 SPRACHLICHE KONVENTIONEN

In diesem e-Book werden für sämtliche Komponenten und Features in Power BI die englischen Begriffe verwendet. Einerseits deshalb, weil die Übersetzungen in der deutschen Sprachversion teilweise völlig unbrauchbar sind, andererseits weil mit den englischen Originalbegriffen die Webrecherche viel ergiebigere Ergebnisse liefert.

Objektbezeichnungen werden immer in fetter Schriftart gekennzeichnet:

- Aktivieren Sie in der Feldliste jetzt aus der Tabelle **Dim Dates** die Hierarchie **Year Hierarchy**

Menübefehle werden ebenfalls in fetter Schriftart und mit -> dargestellt:

- **Start -> Options and Settings -> Options**
  Das bedeutet: wählen Sie im Menü **Start** aus der Gruppe **Options and Settings** den Befehl **Options**.
- **Kontextmenü des Datenpunkts -> Include**
  Das bedeutet: klicken Sie mit der rechten Maustaste auf einen Datenpunkt (in einem Visual) und wählen den Befehl **Include** aus.

Tastaturbefehle werden so dargestellt:

- [Ctrl] und [+]
  Das bedeutet: drücken Sie bei gehaltener „Strg"-Taste zusätzlich die Taste „+"

DAX-Funktionen werden in Großbuchstaben und mit öffnender/schließender Klammer dargestellt:

- SUM(), ALL(), CALCULATE(), usw.

1.4 POWER BI DESKTOP VERSION

Releasestand: **April 2021**
Sprachversion: **Englisch**

Bitte beachten Sie, daß die Screenshots teilweise aus älteren Programmversionen stammen können, sofern diese sinngemäß auch in der aktuellen Version gültig sind.

1.5 VERWENDETE SHORTCUTS

Hier finden Sie eine kompakte Übersicht über die für dieses e-Book nützlichen Shortcuts für Power BI Desktop:

[Ctrl] + [X]	Markierten Zellinhalt in Zwischenablage **Ausschneiden**
[Ctrl] + [C]	Markierten Zellinhalt in Zwischenablage **Kopieren**
[Ctrl] + [V]	Inhalt der Zwischenablage auf markierte Zelle **Einfügen**
[Ctrl] + [S]	Datei **Speichern**
[Ctrl] + [Z]	Letzten Schritt **rückgängig** machen
[Ctrl] + [Y]	Letzten Schritt **wiederholen**
?	Aufruf der aktuellen Shortcut-Liste

Weiterführend: https://docs.microsoft.com/en-us/power-bi/desktop-accessibility

Folgende Shortcuts sind für die Erstellung von Statements im DAX-Editor („Formelzeile") nützlich:

[Ctrl] + [+] / [-]	**Zoomfaktor** im DAX-Editor vergrößern / verkleinern
[Shift] + [Enter]	**Neue Zeile** (Einrückung behalten)
[Alt] + [Enter]	**Neue Zeile** (ohne Einrückung)
[Tab]	Zeile **nach rechts** einrücken
[Backspace]	Zeile **nach links** einrücken
[Shift] + [Tab]	Zeile **nach links** einrücken
[Alt] + [Pfeil rauf]	Zeile **nach oben** verschieben
[Alt] + [Pfeil runter]	Zeile **nach unten** verschieben
[Ctrl] + [D]	**Suche** des markierten Strings (mehrfach Drücken -> alle markiert -> überschreiben)
[Ctrl] + [Shift] + [L]	**Suchen und Ersetzen** des markierten Strings (markiert alle -> überschreiben)

Weiterführend: https://docs.microsoft.com/en-us/power-bi/desktop-formula-editor

2 POWER BI GRUNDLAGEN

2.1 WAS IST POWER BI?

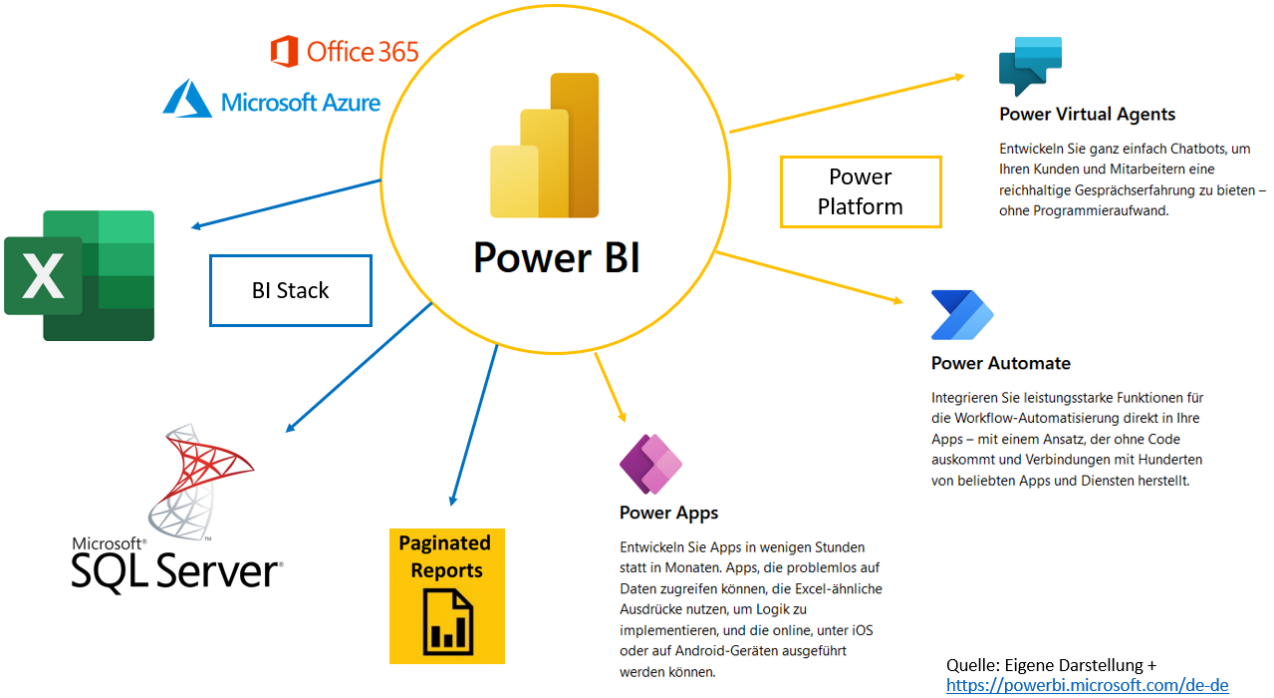
Power BI ist ein cloud-basiertes Dashboarding Tool mit einer sehr leistungsfähigen Query- und Datenmodell-Komponente als „Engine". In Power BI werden sogenannte analytische Datenmodelle aus verschiedenen Datenquellen aufgebaut, die Auswertung erfolgt – vergleichbar mit Power Point – auf einem Canvas.

Was es ist:

Was es nicht ist:

2.2 POWER BI IM KONTEXT DES MICROSOFT PRODUKTPORTFOLIOS

Power BI ist ein Cloud Service, der im Hintergrund auf Microsoft Azure Plattform gehostet wird und der eng mit dem Service Office 365 verzahnt ist. Power BI ist im Kontext des (älteren) **BI Stacks** sowie der (neueren) **Power Platform** zu sehen, die Technologien dieser Produktfamilien ergänzen sich gegenseitig:



Quelle: Eigene Darstellung + https://powerbi.microsoft.com/de-de

1. **Power BI** ist das Tool für Dashboarding und mobiles Reporting
2. **Excel** ist das Tool für die adhoc Analyse und Simulation.

   Natürlich wird Excel in der Praxis auch intensivst für das (Management) Reporting sowie für die Planung/Forecasting verwendet.
3. **SQL Server** ist das Tool für die Realisierung von Enterprise BI Anforderungen.
4. **Paginated Reports** ist das Tool zur Realisierung von Standardberichten im PDF- und Druckformat.
5. **Power Apps** ist das Tool für die Datenerfassung

   In Power BI gibt ein eigenes Visual für die Integration von Power Apps Erfassungsformularen, in Power Apps gibt es ein eigenes Set an Funktionen für die Interaktion mit Power BI Dashboards. Bitte beachten Sie, daß Power Apps kein Planungstool ist sondern ein Tool zur Erstellung von Erfassungsformularen.
6. **Power Automate** ist das Tool für die Prozessautomatisierung

   In Power Automate gibt es sowohl Power BI Inputs (= Auslöser) als auch Power BI Outputs (= Ausgabe), die bspw. genutzt werden können, um automatisch E-Mails mit Power BI Reports zu versenden.
7. **Power Virtual Agents** ist das Tool zur Erstellung von Bots.

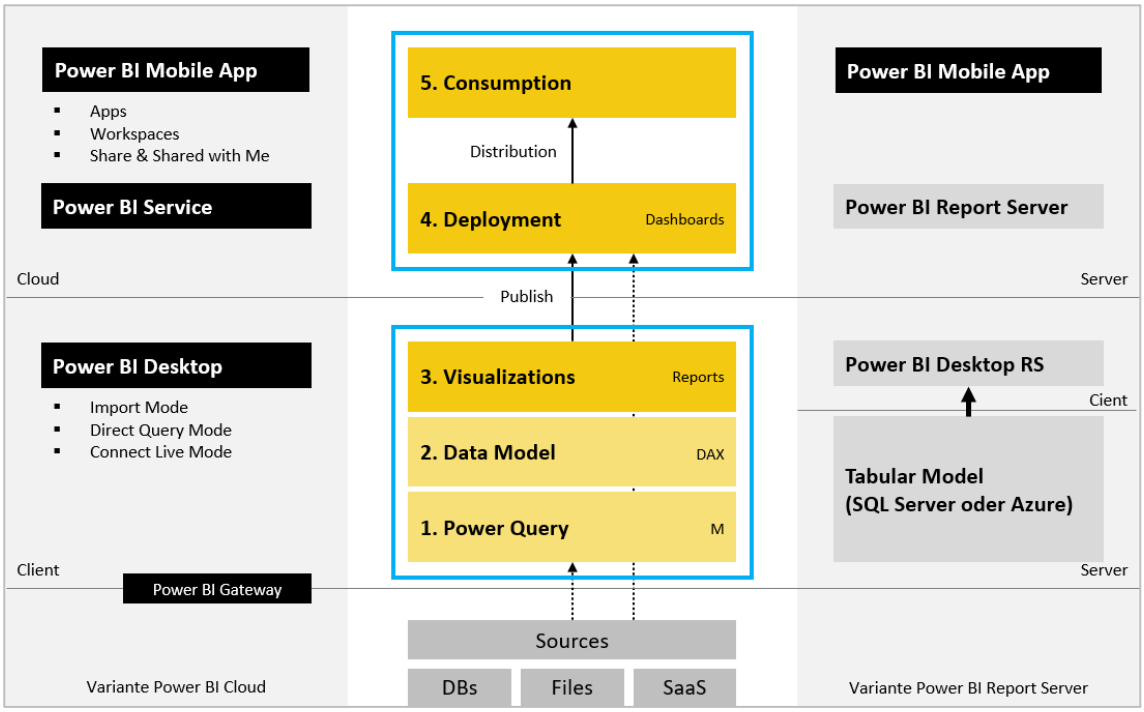
Die Technologien **Power Query** (M-Formelsprache) und das **Datamodel** (VertiPaq/Direct Query, Beziehungen & DAX-Formelsprache) sind nicht nur in Power BI sondern auch in Excel und SQL Server Tabular Model verfügbar. Damit kann das 1x aufgebaute Know-How rasch auf die anderen Produkte angewendet werden.

Diese Technologien haben in den 3 Produkten einige (sinnvolle) Spezifika, unterscheiden sich unglücklicherweise aber auch im Entwicklungsstand. Excel Power Pivot wird seit einigen Jahren nicht mehr weiterentwickelt, während das Power BI Datamodel und das SQL Server Tabular Model sich auf einem hohen (aber trotzdem nicht ganz identen) Entwicklungsstand befinden.

2.3 POWER BI ARCHITEKTUR UND FOKUS DIESES E-BOOKS

Power BI entfaltet sich aus dem Zusammenspiel von 3 Programmen – Power BI Desktop, Power BI Service, Power BI Mobile App – die insgesamt 5 Funktionen erfüllen:

1. **Power Query** (Extraktion, Transformation, Laden, M-Funktionen)
2. **Data Model** (Analytische Datenbank, Filterbeziehungen, Measures, DAX-Funktionen)
3. **Visualisierung** (visuelle Aufbereitung der KPIs, Interaktivität)
4. **Deployment** (24-Stunden Serverbetrieb, Berechtigungen, Aktualisierung)
5. **Consumption** (einfacher Zugang für Management und Information Worker)

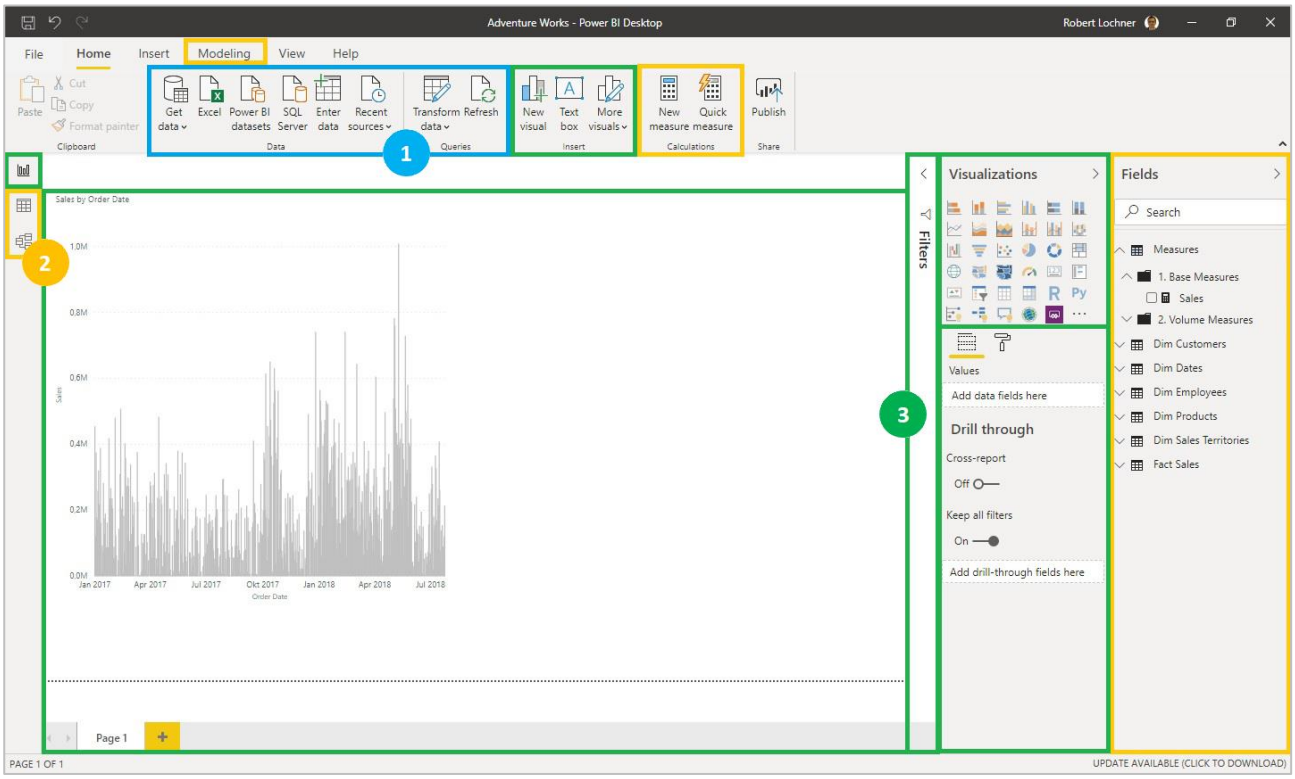


Der Fokus dieses e-Books liegt auf der Komponente „2. Data Model".

2.4 USER INTERFACE VON POWER BI DESKTOP

Die 3 Architekturkomponenten können im User Interface von Power BI Desktop folgendermaßen zugeordnet werden:

1. **Get Data / Queries** („blau")
2. **Datamodel** („orange")
3. **Visualisierung** („grün")



2.5 ZENTRALE BEGRIFFE IM DATAMODEL

Datenmodell | Datamodel
= Sammlung der Tabellen, Spalten, Beziehungen, Measures und (Row Level) Security in einer Power BI Anwendung

Tabelle | Table
= Tabelle in einem Datenmodell (diese kann aus mehreren Quelltabellen zusammengesetzt worden sein)
- Faktentabelle (Foreign Keys, Wertspalten, Faktenattribute)
- Dimensionstabelle (Primary Keys und Attribute)
- Bridge Tabelle
- Parametertabelle

Spalte / Feld / Attribut | Column / Field / Attribute
= Spalte einer Tabelle (= Datum, Text, Zahl, Image, u.a.)

Zeile / Datensatz | Row / Record
= Zeile einer Tabelle

(Verschiedene) Elemente | (Distinct) Values
= (verschiedene) Einträge in einer Spalte einer Tabelle

Beziehungen | Relationships
= Verknüpfungen zwischen Tabellen (mit bestimmter Granularität und Filterrichtung)

Filterweitergabe | Filter Propagation
= Weitergabe eines Filters von einer Tabelle auf eine verbundene Tabelle über eine oder mehrere Beziehungen

Measure | Measure
= mit DAX berechnete (und häufig gefilterte) Aggregation auf eine oder mehrere Spalten des Datenmodells

Berechnete Spalte | Calculated Column
= mit DAX berechnete Spalte, idR als Ableitung einer bestehenden Spalte

Berechnete Tabelle | Calculated Tables
= mit DAX berechnete Tabelle, häufig als Ableitung einer bestehenden Tabelle

Berechtigungssystem | Row Level Security
= mit DAX erstellte Bedingung, welche Datensätze für welche User angezeigt werden dürfen

DAX Abfragen | DAX Queries
= mit DAX erstellte Abfrage auf das Datamodel „von außen"

DAX | DAX
= Data Analysis Expressions
= Formelsprache für Measures, Calculated Columns und Calculated Tables

3 DAX GRUNDLAGEN

3.1 EINSTIEG IN DIE DAX FORMELSPRACHE

3.1.1 AUFGABENSTELLUNG

1 Einfache Aufgabe

Aus dem folgenden Datenmaterial in Excel soll folgendes ermittelt werden:

- Absatz und Umsatz
- Gesamt und je Artikel

Adventure Works (AW) Reseller
Sales Table

Article Key	Employee Key	Order Date	Order Number	Order Quantity
A262	E285	03.10.2017	SO43685	1
A262	E281	01.10.2017	SO43688	1
A221	E281	06.10.2017	SO46620	2
A224	E281	06.10.2017	SO46620	4
A224	E282	06.10.2017	SO46625	6
A221	E282	06.10.2017	SO46625	10
A224	E281	03.10.2017	SO46633	3
A221	E281	03.10.2017	SO46633	6
A224	E283	03.10.2017	SO50258	2
A224	E283	03.10.2017	SO50259	2
A221	E283	03.10.2017	SO50259	5
A224	E285	01.10.2017	SO50279	2
A224	E281	06.10.2017	SO50289	4
A221	E281	06.10.2017	SO50289	5
A224	E282	03.10.2017	SO50294	2
A221	E282	03.10.2017	SO50294	4
A221	E283	01.10.2017	SO50311	2
A224	E283	01.10.2017	SO50311	10

Article Table

Article Key	Article Name	Product Subcategory	Product Category	Standard Cost	List Price
A221	Sport-100 Helmet, Blue	Helmets	Accessories	13,8782	35,0442
A224	AWC Logo Cap	Caps	Clothing	5,2297	8,6442
A262	LL Road Frame - Red, 44	Road Frames	Components	181,4857	306,5636
A265	LL Road Frame - Red, 48	Road Frames	Components	187,1571	337,2200

Power BI Camp DAX mit Power BI

3.1.2 LÖSUNG IN EXCEL – SUMME(), SVERWEIS() UND SUMMEWENN()

Die Aufgabe wird in Excel mit Formeln in Zellen gelöst – wir nennen dieses Lösungsinstrument hier „Formelzellen". N den Formelzellen kommen die Funktionen SUMME() und SVERWEIS() …

2 Lösung in Excel

Komponenten
- **Basis = Sheets + Datenzellen**
- **Formelzellen**

Konzepte für Formeln in Zellen:
1. Eine Formel liefert **genau 1 Ergebnis**
2. Die Evaluierung (= Berechnung) der Excel Formeln erfolgt im **Cell-Context**, d.h. ändert sich der Inhalt der Quellzellen ändert sich auch das Ergebnis.

Adventure Works (AW) Reseller
Sales Table

Article Key	Employee Key	Order Date	Order Number	Order Quantity	Sales Price	Sales Amount
A262	E285	03.10.2017	SO43685	1	306,5636	306,56
A262	E281	01.10.2017	SO43688	1	306,5636	306,56
A221	E281	06.10.2017	SO46620	2	33,6442	67,29
A224	E281	06.10.2017	SO46620	4	8,6442	34,58
A224	E282	06.10.2017	SO46625	6	8,6442	51,87
A221	E282	06.10.2017	SO46625	10	33,6442	336,44
A224	E281	03.10.2017	SO46633	3	8,6442	25,93
A221	E281	03.10.2017	SO46633	6	33,6442	201,87
A224	E283	03.10.2017	SO50258	2	8,6442	17,29
A224	E283	03.10.2017	SO50259	2	8,6442	17,29
A221	E283	03.10.2017	SO50259	5	33,6442	168,22
A224	E285	01.10.2017	SO50279	2	8,6442	17,29
A224	E281	06.10.2017	SO50289	4	8,6442	34,58
A221	E281	06.10.2017	SO50289	5	33,6442	168,22
A224	E282	03.10.2017	SO50294	2	8,6442	17,29
A221	E282	03.10.2017	SO50294	4	33,6442	134,58
A221	E283	01.10.2017	SO50311	2	33,6442	67,29
A224	E283	01.10.2017	SO50311	10	8,6442	86,44
			Total:	71		2.059,58

Article Table

Article Key	Article Name	Product Subcategory	Product Category	Standard Cost	List Price
A221	Sport-100 Helmet, Blue	Helmets	Accessories	13,8782	33,6442
A224	AWC Logo Cap	Caps	Clothing	5,2297	8,6442
A262	LL Road Frame - Red, 44	Road Frames	Components	181,4857	306,5636
A265	LL Road Frame - Red, 48	Road Frames	Components	187,1571	337,2200

Excel Formeln zur Ermittlung der Gesamtergebnisse:
- F25= SUMME(F6:F23)
- G6= SVERWEIS(B6;$J$6:$O$9;6)
  Auskopieren bis Zeile 23
- H6= F6*G6
  Auskopieren bis Zeile 23
- H25= SUMME(H6:H23)

Power BI Camp DAX mit Power BI

… sowie SUMMEWENN() zum Einsatz:

2 Lösung in Excel

Komponenten
- **Basis = Sheets + Datenzellen**
- **Formelzellen**

Konzepte für Formeln in Zellen:
1. Eine Formel liefert **genau 1 Ergebnis**
2. Die Evaluierung (= Berechnung) der Excel Formeln erfolgt im **Cell-Context**, d.h. ändert sich der Inhalt der Quellzellen ändert sich auch das Ergebnis.

Adventure Works (AW) Reseller
Sales Table

Article Key	Employee Key	Order Date	Order Number	Order Quantity	Sales Price	Sales Amount
A262	E285	03.10.2017	SO43685	1	306,5636	306,56
A262	E281	01.10.2017	SO43688	1	306,5636	306,56
A221	E281	06.10.2017	SO46620	2	33,6442	67,29
A224	E281	06.10.2017	SO46620	4	8,6442	34,58
A224	E282	06.10.2017	SO46625	6	8,6442	51,87
A221	E282	06.10.2017	SO46625	10	33,6442	336,44
A224	E281	03.10.2017	SO46633	3	8,6442	25,93
A221	E281	03.10.2017	SO46633	6	33,6442	201,87
A224	E283	03.10.2017	SO50258	2	8,6442	17,29
A224	E283	03.10.2017	SO50259	2	8,6442	17,29
A221	E283	03.10.2017	SO50259	5	33,6442	168,22
A224	E285	01.10.2017	SO50279	2	8,6442	17,29
A224	E281	06.10.2017	SO50289	4	8,6442	34,58
A221	E281	06.10.2017	SO50289	5	33,6442	168,22
A224	E282	03.10.2017	SO50294	2	8,6442	17,29
A221	E282	03.10.2017	SO50294	4	33,6442	134,58
A221	E283	01.10.2017	SO50311	2	33,6442	67,29
A224	E283	01.10.2017	SO50311	10	8,6442	86,44
			Total:	71		2.059,58

Article Table

Article Key	Article Name	Product Subcategory	Product Category	Standard Cost	List Price
A221	Sport-100 Helmet, Blue	Helmets	Accessories	13,8782	33,6442
A224	AWC Logo Cap	Caps	Clothing	5,2297	8,6442
A262	LL Road Frame - Red, 44	Road Frames	Components	181,4857	306,5636
A265	LL Road Frame - Red, 48	Road Frames	Components	187,1571	337,2200

Article	Quantity	Sales
A262	2	613,13
A221	34	1.143,90
A224	35	302,55
Total	71	2.059,58

Excel Formeln zur Ermittlung der Teilergebnisse:
- M13= SUMMEWENN($B$6:$B$23;$L13;$F$6:$F$23)
  Auskopieren bis Zeile 15
- N13= SUMMEWENN($B$6:$B$23;$L13;$H$6:$H$23)
  Auskopieren bis Zeile 15
- M16= SUMME(M13:M15)
- N16= SUMME(N13:N15)

Power BI Camp DAX mit Power BI

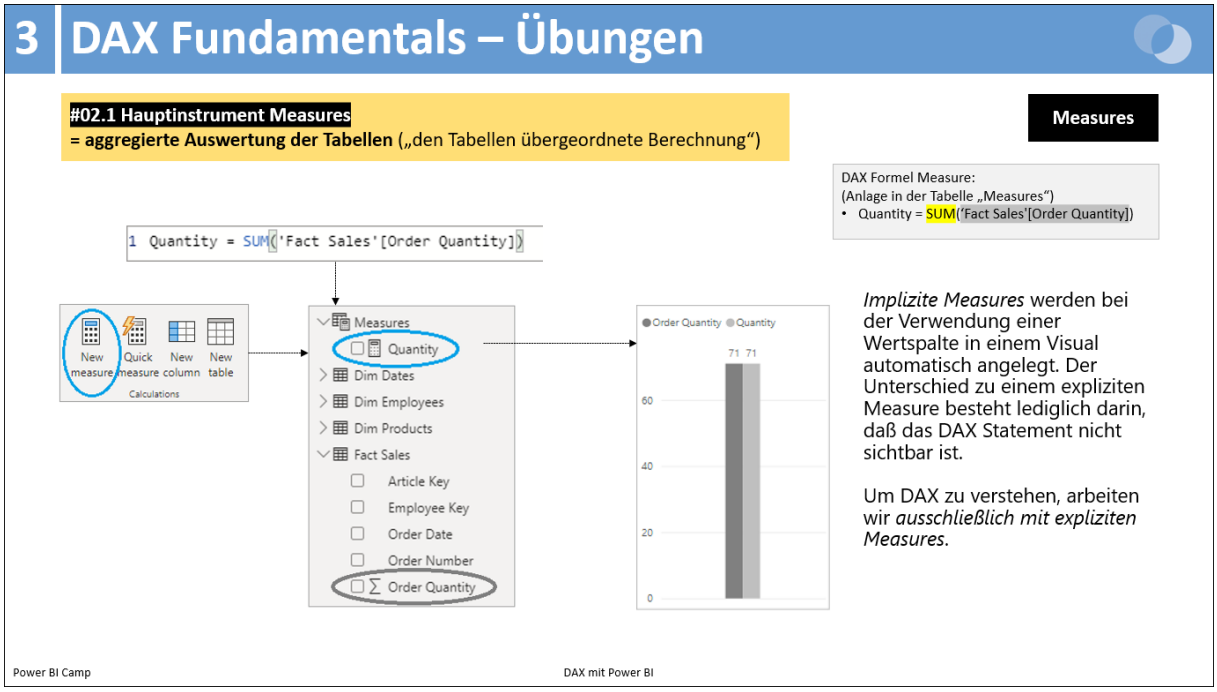
3.1.3 LÖSUNG IN POWER BI – ERARBEITUNG DER DAX FUNDAMENTALS

In Power BI werden **sämtliche Daten in Tabellen gespeichert**, die über **Filterbeziehungen** verbunden sein können (aber nicht müssen). Das vorliegende Beispiel kann als „Fact Sales" und „Dim Products" in Power BI tabellarisch abgebildet werden, die beiden Tabellen sind über eine Filterbeziehung zu einem Datenmodell verbunden:

Die Ermittlung der Gesamtmenge erfolgt in Power BI mit einem **Measure**:

```
Quantity = SUM('Fact Sales'[Order Quantity])
```

Das hat Ähnlichkeiten mit einer Formelzelle, ist aber etwas abstrakter als in Excel. Das Measure mit der DAX Formel wird im Datamodel definiert, das Berechnungsergebnis ist dort aber nicht zu sehen. Die Evaluierung (= Berechnung) eines Measures erfolgt erst zur Laufzeit bei der Verwendung in einem Visual, die Berechnungsergebnisse des Measures werden daher erst in den Visuals angezeigt:

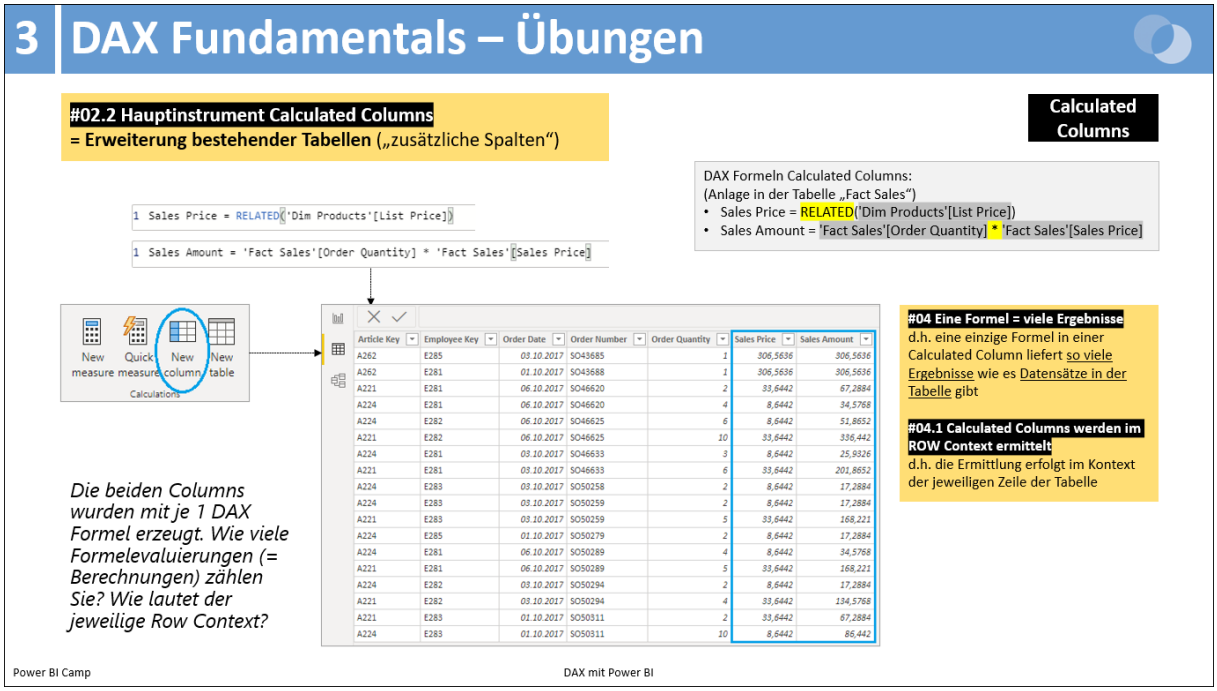


Zur Ermittlung des Gesamtumsatzes werden in Power BI (ähnlich wie in Excel) zwei **Calculated Columns** angelegt, um den **List Price** aus der Dimensionstabelle in die Faktentabelle zu holen und dort mit der **Quantity** zur **Sales Amount** zu multiplizieren:

```
Sales Price = RELATED('Dim Products'[List Price])
```

```
Sales Amount = 'Fact Sales'[Order Quantity] * 'Fact Sales'[Sales Price]
```

Die Berechnungsergebnisse der Calculated Columns werden unmittelbar in der Datenansicht des Datamodels der jeweiligen Tabelle angezeigt:



Article Key	Employee Key	Order Date	Order Number	Order Quantity	Sales Price	Sales Amount
A262	E285	03.10.2017	SO43685	1	306,5636	306,5636
A262	E281	01.10.2017	SO43688	1	306,5636	306,5636
A221	E281	06.10.2017	SO46620	2	33,6442	67,2884
A224	E281	06.10.2017	SO46620	4	8,6442	34,5768
A224	E282	06.10.2017	SO46625	6	8,6442	51,8652
A221	E282	06.10.2017	SO46625	10	33,6442	336,442
A224	E281	03.10.2017	SO46633	3	8,6442	25,9326
A221	E281	03.10.2017	SO46633	6	33,6442	201,8652
A224	E283	03.10.2017	SO50258	2	8,6442	17,2884
A224	E283	03.10.2017	SO50259	2	8,6442	17,2884
A221	E283	03.10.2017	SO50259	5	33,6442	168,221
A224	E285	01.10.2017	SO50279	2	8,6442	17,2884
A224	E281	06.10.2017	SO50289	4	8,6442	34,5768
A221	E281	06.10.2017	SO50289	5	33,6442	168,221
A224	E282	03.10.2017	SO50294	2	8,6442	17,2884
A221	E282	03.10.2017	SO50294	4	33,6442	134,5768
A221	E283	01.10.2017	SO50311	2	33,6442	67,2884
A224	E283	01.10.2017	SO50311	10	8,6442	86,442

Hinweis: die Column **Sales Amount** wird anschließend wiederum mit einem (weiteren) Measure aggregiert ausgewertet (hier nicht abgebildet).

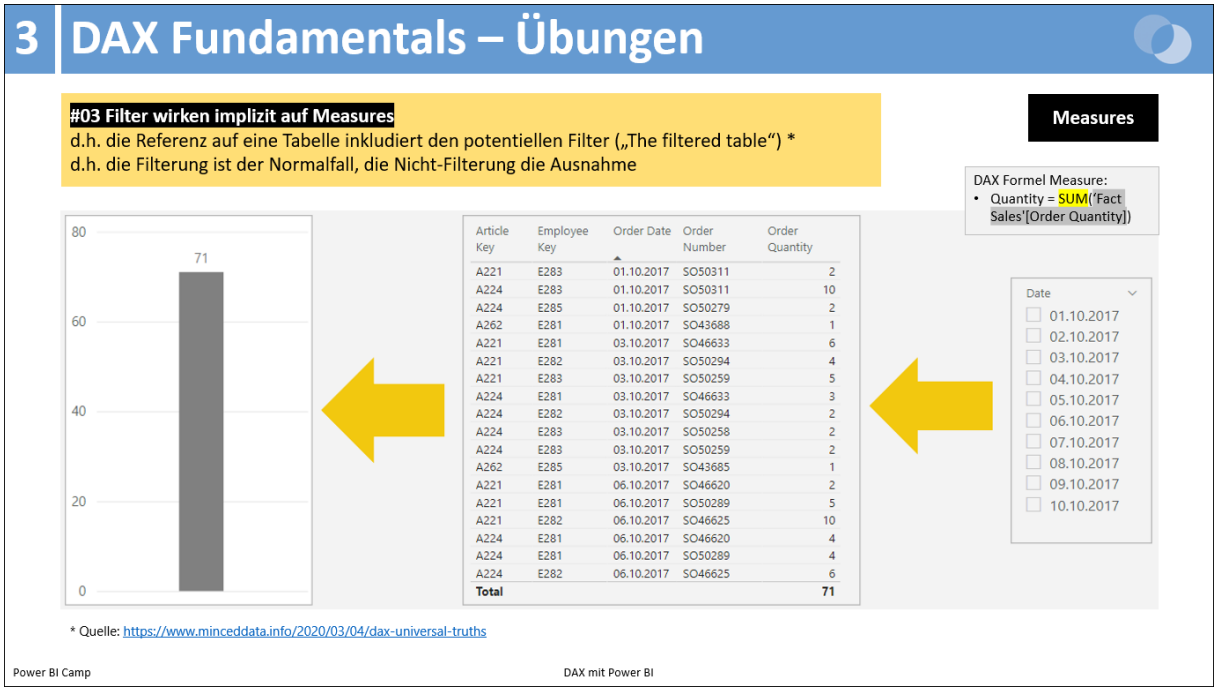
Konzepte für Calculated Columns:

a. Eine einzige Formel liefert so **viele Ergebnisse**, wie es Datensätze in der Tabelle gibt.
b. Die Evaluierung (= Berechnung) einer Calculated Column erfolgt im **Row Context**, d.h. die Ermittlung erfolgt im Kontext der jeweiligen Zeile der Tabelle.

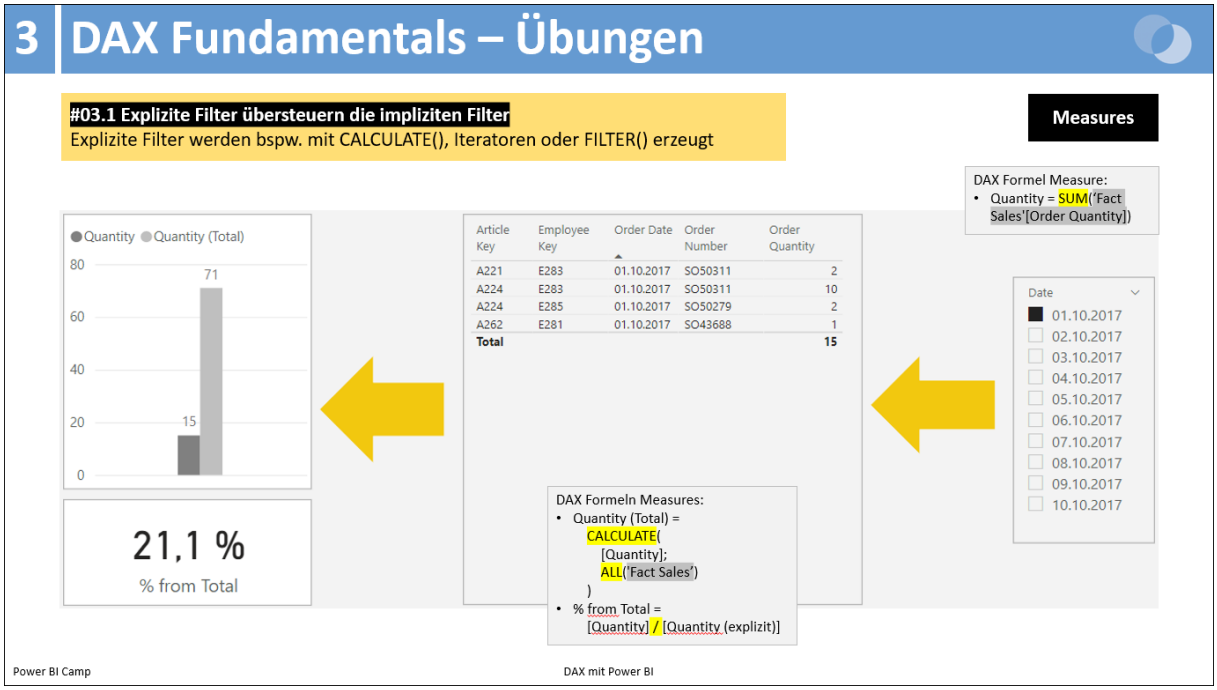
   Frage: wie oft wurden die beiden DAX-Formeln der Calculated Column im Screenshot oben evaluiert?

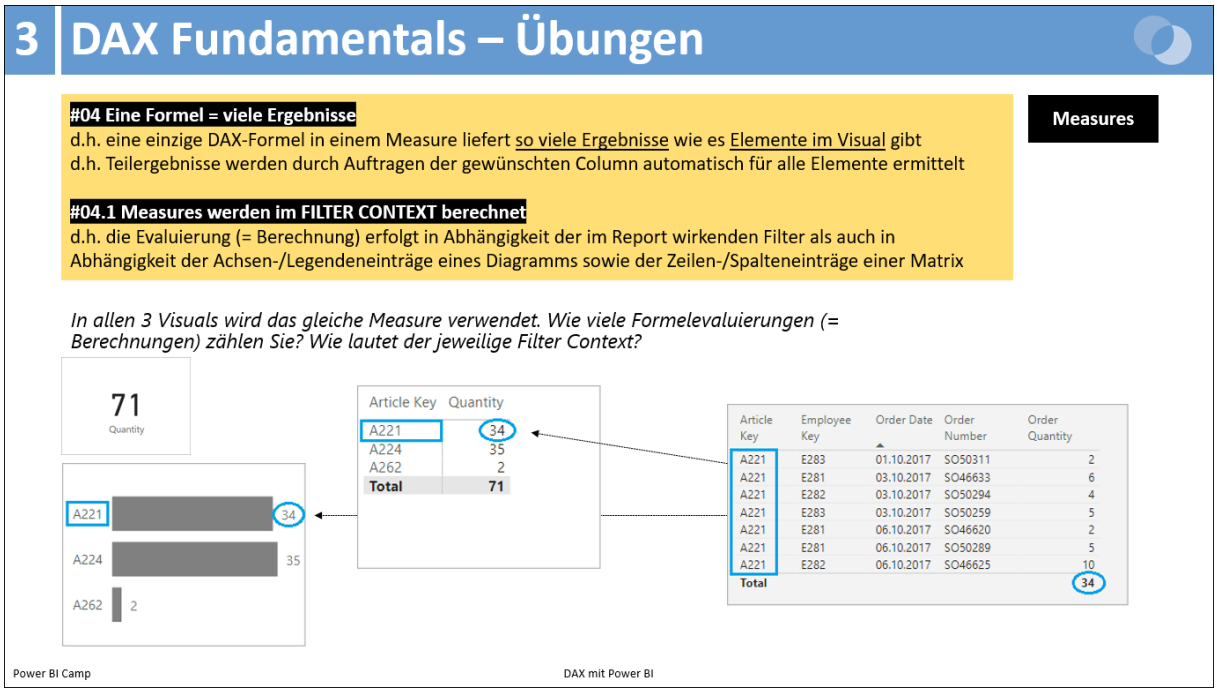
Zurück zu den Measures.

Measures werden implizit (d.h. „grundsätzlich immer") von den anderen Visuals einer Berichtsseite in Power BI gefiltert:



Explizite Filter können – bspw. mit Hilfe der DAX-Funktion CALCULATE() – definiert werden und übersteuern dann die implizite Filterung. Im Vergleich zu Excel ist das genau das gegenteilige Verhalten:





Konzept:

1. Eine einzige Formel liefert so **viele Ergebnisse** wie es Knotenpunkte in den Visuals einer Berichtsseite gibt
2. Die Evaluierung (= Berechnung) eines Measures erfolgt im **Filter Context**, d.h. in Abhängigkeit der im Report wirkenden Filter als auch in Abhängigkeit der Achsen-/Legendeneinträge eines Diagramms sowie Zeilen-/Spalteneinträge einer Matrix.

   Frage: wie oft wurde die DAX-Formel des Measures im Screenshot oben evaluiert?

Aus diesem Grundprinzip folgen zwei sehr weitreichende Konsequenzen für das Verhalten von Measures:

3 DAX Fundamentals – Übungen

#04.2 Die TOTAL Ebene im Visual wird immer eigenständig berechnet
d.h. die TOTAL Zeile ist nicht die Summe der darüber liegenden Zeilen im Visual sondern wird völlig unabhängig in ihrem eigenen Filter Context berechnet
d.h. auch, daß es in DAX keine Möglichkeit gibt, in einer Formel auf ein im Visual "benachbartes Ergebnis" zu referenzieren (die Lösung solcher Aufgaben erfolgt durch Herstellung des gewünschten Filter Context)

Measures

Das ist eine logische Folge des „Eine Formel = viele Ergebnisse“ Prinzips und ist wichtig insbesondere für
1. die Berechnung in Visuals ohne Detailzeilen
2. die Berechnung nicht additiver Measures

DAX Formeln Measures:
- Number of Sales = DISTINCTCOUNT('Fact Sales'[Order Number])
- Filter Context (Dim) = CONCATENATEX(
  'Dim Products';
  'Dim Products'[Article Key];
  ";
)

Article Key	Quantity	Number of Sales
A221	34	7
A224	35	9
A262	2	2
Total	**71**	**11**

71
Quantity

Sichtbarmachen Filter Context

Article Key	Filter Context (Dim)
A221	A221
A224	A224
A262	A262
A265	A265
Total	**A221; A224; A262; A265**

Power BI Camp DAX mit Power BI

3 DAX Fundamentals – Übungen

#04.3 Ein BLANK() Berechnungsergebnis unterdrückt das Element im Visual
d.h. alle Visuals in Power BI zeigen nur diejenigen Elemente einer Column an, für die das Berechnungsergebnis <> BLANK() ist

Measures

Article Key
A221
A224
A262
A265

Article Key	Quantity
A221	34
A224	35
A262	2
Total	**71**

Article Key	Quantity	Demo BLANK
A221	34	yes
A224	35	yes
A262	2	
Total	**71**	

Article Key	Demo BLANK
A221	yes
A224	yes
Total	

Article Key	Demo BLANK	Demo Non-BLANK
A221	yes	yes
A224	yes	yes
A262		no
A265		no
Total		**no**

DAX Formeln Measures:
- Demo BLANK = IF(
  SELECTEDVALUE('Dim Products'[Article Key]) IN {"A221";"A224"};
  "yes";
  BLANK()
)
- Demo Non-BLANK = IF(
  SELECTEDVALUE('Dim Products'[Article Key]) IN {"A221";"A224"};
  "yes";
  "no"
)

Power BI Camp DAX mit Power BI

Die Frage, wann Measures und wann Calculated Columns erstellt werden sollen, ist auch in fortgeschrittenen Lösungsaufgaben eine sehr zentrale Vorfrage.

3 DAX Fundamentals – Übungen

#05 Measures vs. Columns

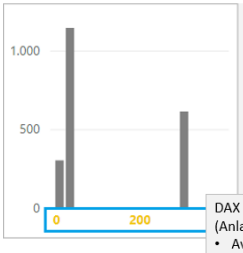
Measures + Columns

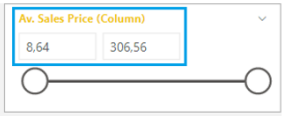
Measures für Aggregationen
- Ausnahmsweise auf Visual-Ebene auch als Filter einsetzbar
- Berechnung kann zur Laufzeit verändert werden

Columns für Achsen und Filter
- Gilt gleichermaßen für importierte Columns wie Calculated Columns
- Berechnung kann nicht zur Laufzeit verändert werden

Article Key	Quantity	Sales	Av. Sales Price (Measure)
A221	34	1.143,90	33,64
A224	35	302,55	8,64
A262	2	613,13	306,56
Total	**71**	**2.059,58**	**29,01**

DAX Formeln Measures:
- Sales = SUM('Fact Sales'[Sales Amount])
- Av. Sales Price (Measure) =
```
DIVIDE(
    [Sales];
    [Quantity]
)
```





Date	8,64	33,64	306,56	Total
01.10.2017	12	2	1	15
03.10.2017	9	15	1	25
				31
				71

DAX Formeln Calculated Columns:
(Anlage in der Tabelle „Fact Sales“)
- Av. Sales Price (Column) =
```
DIVIDE(
    'Fact Sales'[Sales Amount];
    'Fact Sales'[Order Quantity]
)
```

Power BI Camp DAX mit Power BI

Die beiden Calculated Columns, die zuvor in der Faktentabelle angelegt wurden, brauchen bei Verwendung einer sogenannten **Iterator Funktion** (in einem Measure) gar nicht angelegt werden sondern können auch virtuell in einem Measure erzeugt werden:

3 DAX Fundamentals – Übungen

#06 Iteratoren = Calculated Column Funktionalität für Measures
d.h. Calculated Column(s) können auch virtuell innerhalb einer Iterator Funktion – bspw. SUMX() – erzeugt werden und müssen daher nicht physisch in der Tabelle angelegt werden

Measures

Fact Sales

Article Key	Employee Key	Order Date	Order Number	Order Quantity	Sales Price	Sales Amount
A221	E283	01.10.2017	SO50311	2	33,64	67,29
A224	E283	01.10.2017	SO50311	10	8,64	86,44
A224	E285	01.10.2017	SO50279	2	8,64	17,29
A262	E281	01.10.2017	SO43688	1	306,56	306,56
A221	E281	03.10.2017	SO46633	6	33,64	201,87
A221	E282	03.10.2017	SO50294	4	33,64	134,58
A221	E283	03.10.2017	SO50259	5	33,64	168,22
A224	E281	03.10.2017	SO46633	3	8,64	25,93
A224	E282	03.10.2017	SO50294	2	8,64	17,29
A224	E283	03.10.2017	SO50258	2	8,64	17,29
A224	E283	03.10.2017	SO50259	2	8,64	17,29
A262	E285	03.10.2017	SO43685	1	306,56	306,56
A221	E281	06.10.2017	SO46620	2	33,64	67,29
A221	E281	06.10.2017	SO50289	5	33,64	168,22
A221	E282	06.10.2017	SO46625	10	33,64	336,44
A224	E281	06.10.2017	SO46620	4	8,64	34,58
A224	E281	06.10.2017	SO50289	4	8,64	34,58
A224	E282	06.10.2017	SO46625	6	8,64	51,87

Article Key	Sales (Iterator)
A221	1.143,90
A224	302,55
A262	613,13
Total	**2.059,58**

DAX Formel Measure:
- Sales (Iterator) =
```
SUMX(
    'Fact Sales';
    'Fact Sales'[Order Quantity] * RELATED('Dim Products'[List Price])
)
```

Power BI Camp DAX mit Power BI

Abschließend: mit dem Konzept der **Context Transition** kann der Filter Context auch in Calculated Column genutzt werden und damit zahlreiche Praxisfälle gelöst werden:

3 DAX Fundamentals – Übungen

#07 Context Transition = Filter Context für Calculated Columns
d.h. Calculated Columns haben per Default nur den Row Context und keinen Filter Context
Der CALCULATE()-Wrapper bzw. die Referenz auf ein Measures bewerkstelligen die Context Transition

Measures + Columns

DAX Formeln Calculated Columns:
(in der Tabelle „Dim Products")
- Quantity Sold = SUM('Fact Sales'[Order Quantity])
- Quantity Sold (with CT) =
```
CALCULATE(
    SUM('Fact Sales'[Order Quantity])
)
```

Article Key	Article Name	Product Category	Product Subcategory	Standard Cost	List Price	Quantity Sold	Quantity Sold (with CT)
A221	Sport-100 Helmet, Blue	Accessories	Helmets	13,878	33,644	71	34
A224	AWC Logo Cap	Clothing	Caps	5,230	8,644	71	35
A262	LL Road Frame - Red, 44	Components	Road Frames	181,486	306,564	71	2
A265	LL Road Frame - Red, 48	Components	Road Frames	187,157	337,220	71	

DAX Formel Measure:
(in der Tabelle „Dim Products")
- Average Quantity per Article (with CT) =
```
AVERAGEX(
    VALUES('Dim Products'[Article Key]);
    CALCULATE(AVERAGE('Fact Sales'[Order Quantity]))
)
```

Article Key	Average of Order Quantity
A221	4,86
A224	3,89
A262	1,00
Total	**3,94**

Article Key	Average Quantity per Article (with CT)
A221	4,86
A224	3,89
A262	1,00
Total	**3,25**

Power BI Camp DAX mit Power BI

Zusammengefasst ergeben sich folgende **DAX Fundamentals**, diese Leitsätze helfen uns dabei, komplexe Fragestellungen zu lösen bzw. DAX Lösungsbausteine rasch erklären zu können:

3 DAX Fundamentals – Zusammenfassung

#01 Datenbasis = Tabellen + Beziehungen
- **#01.1 Filter Propagation**
- **#01.2 Blank Row**

#02 Hauptinstrumente
- **#02.1 Measures** = aggregierte Auswertung der Tabellen
- **#02.2 Calculated Columns** = Erweiterung der Tabellen
- **#02.3 Calculated Tables** = Anlage neuer Tabellen

#03 Filter wirken implizit auf Measures
d.h. die Referenz auf eine Tabelle inkludiert den potentiellen Filter („The filtered table")
- **#03.1 Explizite Filter übersteuern die impliziten Filter**
- **#03.2 Filter in Power BI sind immer Tabellen**

#04 Eine Formel = viele Ergebnisse
- **#04.1 Measures werden im FILTER CONTEXT berechnet, Calculated Columns im ROW CONTEXT**
- **#04.2 Die TOTAL Ebene im Visual wird immer eigenständig berechnet**
- **#04.3 Ein BLANK() Berechnungsergebnis unterdrückt das Element im Visual**

#05 Measures vs. Columns
Measures für Aggregationen, Columns als Achsen und Filter
Measures zur Laufzeit veränderbar, Calculated Columns nicht

#06 Iteratoren = Calculated Column Funktionalität für Measures

#07 Context Transition = Filter Context für Calculated Columns

#08 DAX Variablen
- **#08.1 Outer Filter Context**
- **#08.2 Konvertierung Skalare in Konstanten**

Legende:
Bereits kennengelernt
Lernen wir später

Power BI Camp DAX mit Power BI

3.1.4 CHARAKTERISTIKEN DER DAX FORMELSPRACHE

DAX ist die Formelsprache in Power BI, Analysis Services Tabular Model und Power Pivot. Der Zweck liegt darin, basierend auf den importierten Daten Business Logik aufzubauen. DAX ist nur als englische Sprachversion verfügbar.

DAX hat Ähnlichkeiten mit Excel Formelsprache, daher ist der Einstieg für Excel User relativ einfach. Bei näherer Betrachtung werden die beiden Formelsprachen aber in sehr unterschiedlichen Kontexten ausgeführt, es gelten völlig andere Prinzipien. Werden die unterschiedlichen Prinzipien rasch verinnerlicht, sind auch für Einsteiger bald anspruchsvolle und sehr leistungsfähige DAX Lösungen erreichbar.

	DAX Formeln	Excel Formeln
Beispiele	Sales = SUM('Fact Sales'[Sales Amount]) Sales Amount = 'Fact Sales'[Order Quantity] * RELATED('Dim Product'[List Price])	H25 =SUMME(H6:H23) H6 = F6 * SVERWEIS(B6;$J$6:$N$9;5)
#01 Datenbasis	**Tabellen + Beziehungen**	**Sheets + Datenzellen**
#02 Hauptinstrumente	**Measures** **Calculated Columns**	**Formelzellen**
#03 Filter	**Filter wirken implizit**	**Filter müssen explizit definiert werden**
#04 Formeln	**1 Formel = viele Ergebnisse**	**1 Formel = 1 Ergebnis**
Evaluierungskontext	**Filter Context** **Row Context**	**Cell Context**
Syntax	Referenzen, Funktionen, Operatoren, Konstanten (sehr Excel nahe - englischsprachig)	Referenzen, Funktionen, Operatoren, Konstanten (länderspezifisch)
Ergebnis	Wert, Text, Datum in 1 oder vielen Zellen, einer Spalte oder Tabelle	Wert, Text, Datum in einer Zelle

Abgrenzungen zu SQL und MDX:

- Abgrenzung zu SQL: Beziehungen sind Filterbeziehungen (und keine Constraints), weniger Jointypen, weniger Datentypen, DAX hat Iteratoren and Filter Context in Echtzeit
- Abgrenzung zu MDX: Syntax und Konzepte sind viel simpler, Hierarchien spielen keine Rolle für die DAX Formelsprache

Weiterführend:
https://www.quora.com/Why-do-all-the-BI-consultants-hate-DAX/

3.2 DAX INSTRUMENTE ("BAUSTEIN 1")

Insgesamt existieren 5 Instrumente, in denen DAX Funktionen eingesetzt werden können:

1. Measures
2. Calculated Columns
3. Calculated Tables
4. Row Level Security (RLS)
5. DAX Queries

Relevant für das Power BI Camp sind die ersten drei, diese 3 Hauptinstrumente weisen folgende zentrale Charakteristiken auf:

	MEASURES	Calculated COLUMNS	Calculated TABLES
Beispiel	Sales = SUM('Fact Sales'[Sales Amount])	Sales Price = RELATED('Dim Product'[List Price])	Parameter Tabelle = GENERATESERIES(1;10;1)
Ausführungsebene	"Überhalb" aller Tabellen (Syntax: Aggregationsfunktion zwingend nötig, lediglich bei Iteratoren kann verzichtet werden)	Auf Datensatzebene der "Trägertabelle" (Syntax: direkte Referenz auf Spalten, Aggregationsfunktionen bei Context Transition)	"Neben" den bereits bestehenden Tabellen (Syntax: autonome Erzeugung oder Referenz auf bestehende Tabelle)
Berechnung („Evaluation")	In Echtzeit bei jedem Klick ("dynamische Berechnung")	Beim Processing/Laden ("statische Berechnung")	Beim Processing/Laden ("statische Berechnung")
Wann zu verwenden	**Measure wenn dynamische Berechnung benötigt wird** Jan / Feb / Total Quantity: 28.236 / 23.642 / **51.878** Sales: 16.213.027 / 12.184.295 / **28.397.322**	**Spalte wenn Achse oder Filter benötigt wird** 	**Tabelle wenn ein Auswahlfeld, eine dynamische Dimension, eine Ermittlungsebene, eine Beziehung o.ä. benötigt wird**
Alternativen	> Keine!	> Importierte Spalten > Berechnete Spalten in M	> Importierte Tabellen > Berechnete Tabellen in M
Excel Instrumente	**Formelzellen**		

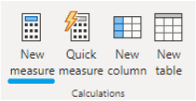
Hinweis: in Excel können Formeln auch in der Bedingten Formatierung, in den Spalten eines Tabellenobjekts sowie in PivotTables/PivotCharts eingegeben werden. Für den Vergleich hier wird aber lediglich das Kerninstrument „Zellen" betrachtet.

3.2.1 INSTRUMENT "MEASURES"

1 Instrument „Measures“

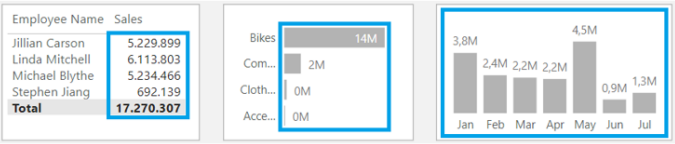
Einsatzgebiete von Measures:

- Dynamische Berechnung im **Werte Bereich** eines Charts, Karte, Matrix oder Tabelle ("Teil- und Gesamtergebnisse")
- Dynamische **Beschriftungen** („Text Measures"), dynamische **Formatierungen** („Farb Measures"), dynamische **Bilder/Indikatoren** („Image Measures"), dynamische **Settings**
- Als **Legende** im Pie-/Donut-Chart, als **Spalten** im Table Visual, als **Spalten oder Zeilen** im Matrix Visual (bei der Auswertung mehrerer Measures)
- Ausnahmsweise als **Visual-Level-Filter**



Eigenschaften:

- Gehören zu keiner konkreten Tabelle
- Werden zur Laufzeit im Kontext der Selektionen des Users – d.h. im **Filter Context** – berechnet und ermöglichen die hohe Interaktivität von Power BI Dashboards („**dynamische Berechnungen**").



DAX Syntax:

- Da die Berechnung „überhalb" der Tabellen erfolgt, kann keine direkte Referenz auf Columns gesetzt werden, sondern müssen Columns **immer mit Aggregationsfunktionen** umschlossen werden (Ausnahme: Iterator-Funktionen)

Power BI Camp DAX mit Power BI

Zusätzliche Eigenschaften:

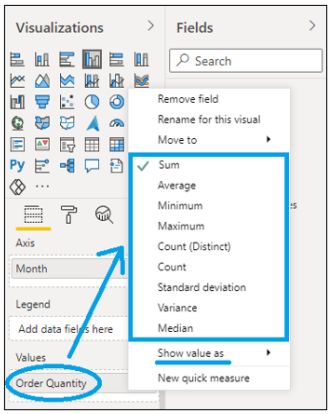
- Benötigen keinen RAM-Speicher jedoch CPU-Rechenleistung.

Automatismen/Wizards:

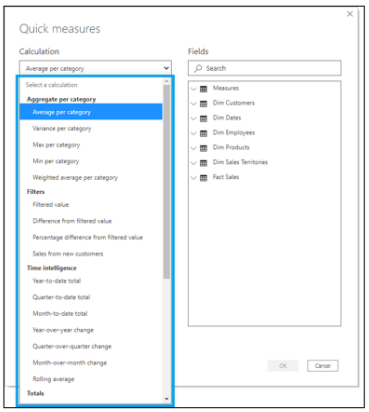
1 Instrument „Measures“

Automatismen/Wizards:

- **Implizite Measures**
  Wertspalten können auch ohne Anlage von „expliziten Measures" mit den gängigen Aggregationsfunktionen ausgewertet werden. Dabei werden auf Basis der getroffenen Selektion – für den User unsichtbar – im Datamodel automatisch Measures angelegt:



- **Quick Measures**
  Für anspruchsvollere Berechnungen stehen die sogenannten Quick Measures zur Verfügung. Für Einsteiger in DAX sind die erzeugten DAX Statement aber häufig unnötig kompliziert und daher nicht immer gut als Lernvorlage geeignet.

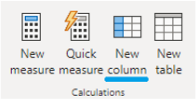


Power BI Camp DAX mit Power BI

3.2.2 INSTRUMENT "CALCULATED COLUMNS"

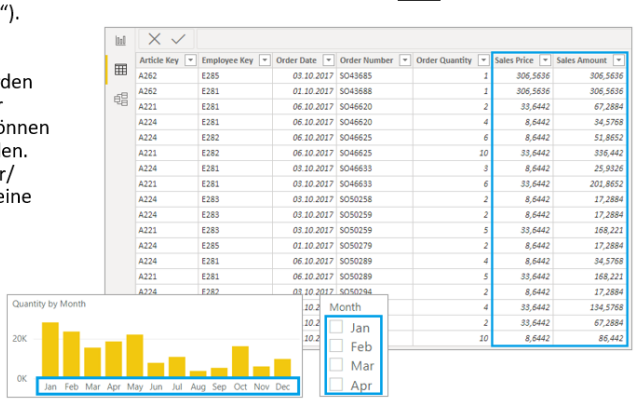
2 Instrument „Calculated Columns“

Einsatzgebiete von Columns:
- Als **Achse/Legende** eines Charts sowie als Zeilen **und/oder** Spalten im Matrix oder Table Visual
- Als **Slicer** sowie als **Filter** (auf allen Ebenen)



Eigenschaften:
- Werden in die gewünschte „Trägertabelle“ eingehängt, in dessen **Row Context** die Berechnung stattfinden soll.
- Werden beim Processing bzw. beim Refresh des Datamodels berechnet, Selektionen des Users können nicht auf die Berechnung einwirken („**statische Berechnungen**“).

DAX Syntax:
- Da die Berechnung „innerhalb“ einer Tabelle erfolgt, werden **direkte Referenzen** auf Columns ohne Verwendung einer Aggregationsfunktion gesetzt. Aggregationsfunktionen können dennoch bei der sog. Context Transition verwendet werden.
- Es gehört übrigens **nicht** zum Konzept, auf den Vorgänger/ Nachfolger-Datensatz („Nachbarzelle“) zuzugreifen, da keine explizite Sortierreihenfolge in einer Tabelle des Power BI Datamodels mitgespeichert wird.



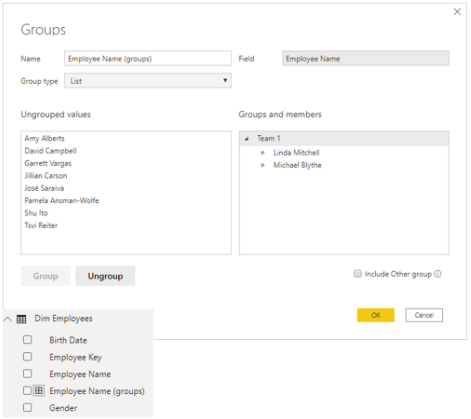
Power BI Camp DAX mit Power BI

Zusätzliche Eigenschaften:
- Werden im Datamodel verspeichert wie importierte Columns und benötigen daher RAM-Speicher.

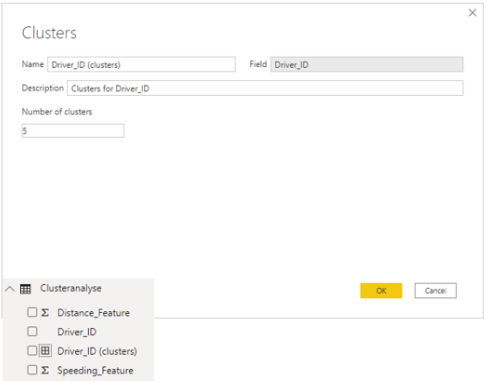
Automatismen/Wizards:

2 Instrument „Calculated Columns“

Automatismen/Wizards:
- **Grouping & Binning**
  Aufruf erfolgt bspw. über Bar Chart



- **Clustering**
  Aufruf erfolgt über das Scatter Chart

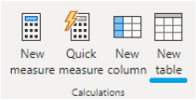


Power BI Camp DAX mit Power BI

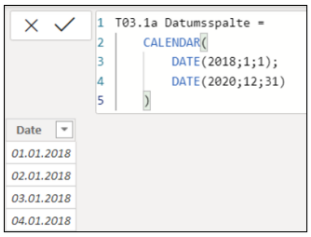
3.2.3 INSTRUMENT "CALCULATED TABLES"

3 Instrument „Calculated Tables“

Einsatzgebiete von Calculated Tables:
- als **Parameter-Tabelle** für Slicer und Selektionen
- als **dynamische Datums- oder Zeitdimension**
- als **dynamische Brückentabelle** zur Erstellung einer **Beziehung** (bspw. im Multi-Fakten Schema)



Eigenschaften:
- Werden beim Processing bzw. beim Refresh des Datamodels berechnet, Selektionen des Users können **nicht** auf die Berechnung einwirken („**statische Berechnungen**“).



DAX Syntax:
- Sehr einfach, wenn kein Bezug auf andere Tabellen genommen wird

Power BI Camp DAX mit Power BI

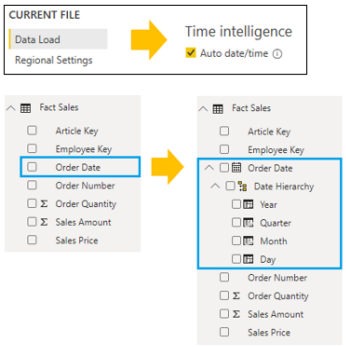
Zusätzliche Eigenschaften:

- Werden im Datamodel gleich verspeichert wie importierte Tabellen und benötigen daher RAM-Speicher.
- Bitte beachten Sie: In Excel Power Pivot gibt es bisher keine Calculated Tables.

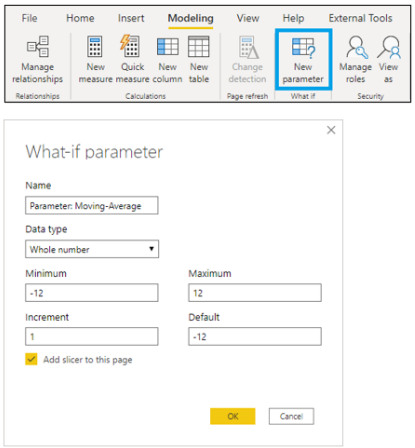
Automatismen/Wizards:

3 Instrument „Calculated Tables“

Automatismen/Wizards:
- **Auto Date Dimension für Datumsfelder**
  Aktivierung über Optionen & Settings



- **What-If Parameter Tabellen**
  Aufruf erfolgt über Button im Menü



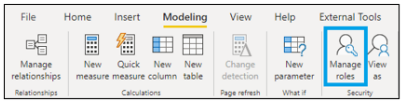
Power BI Camp DAX mit Power BI

3.2.4 INSTRUMENT "ROW LEVEL SECURITY (RLS)"

4 Instrument „Row Level Security“

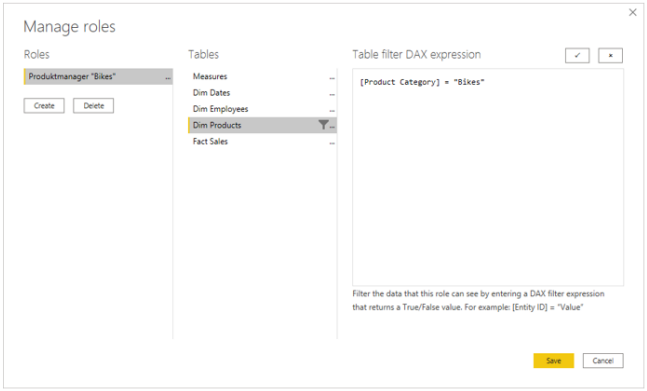
Einsatzgebiete:

- **Einschränkung der Zugriff**sberechtigungen auf bestimmte Datensätze nach dem Muster „Vertriebsleiter Deutschland darf nur Zahlen für Deutschland sehen“ oder „Kostenstellenverantwortlicher darf nur Zahlen der Kostenstelle X sehen“



Eigenschaften:

- Die RLS Funktion verwendet DAX-Statements, um einen „rollenspezifischen Hintergrundfilter“ zu setzen. Die eingesetzten DAX-Statements erzeugen keine Kennzahlen sondern liefern Filterbedingungen, die beim Einstieg eines Users in die Anwendung auf alle Reports der Anwendung angewendet werden.
- Als DAX-Statements kommen daher **nur Filterbedingungen bzw. Tabellenfunktionen** (Filter in DAX sind immer Tabellen) zur Anwendung.



Wizard = User Interface:

- Manage Roles
  Hier können zumindest einfache RLS Patterns mit wenig DAX Know-How erstellt werden.

Siehe weiterführend im Handout „Dashboarding mit Power BI“ (statische und dynamische RLS)

Power BI Camp DAX mit Power BI

Weiterführend:

- https://linearis.at/blog/2018/03/09/berechtigungen-in-power-bi-teil-3-row-level-security-rls/

Bitte beachten Sie:

- In Excel Power Pivot gibt es keine Row Level Security.

3.2.5 INSTRUMENT "DAX QUERIES"

5 Instrument „DAX Queries“

In Power BI gibt es keinen Button zur Anlage einer DAX Query. Der Grund liegt darin, daß für den Aufbau und Betrieb einer normalen Power BI Anwendung keine DAX Queries benötigt werden. DAX Queries werden Power BI intern zur Kommunikation zwischen den Visuals und dem Datamodel verwendet und werden automatisch erzeugt.

Einsatzgebiete von DAX Queries:

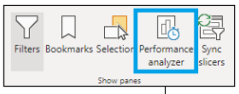
- Abfragen auf das Power BI Datamodel „von außen“mit dem Tool „**DAX Studio**“ zur **Performanceoptimierung**
- Abfragen auf SQL Server Tabular Model (diese sind immer „von außen“) bspw. als **Excel Query Table** (Beispiel hier), mit **SQL Server Reporting Services** oder als SSMS Query
- Definition der **Detail Rows Expresssions** im SQL Server Tabular Model (verfügbar seit SQL Server 2017)

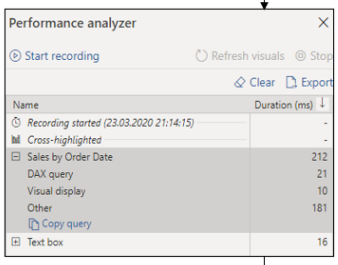
Eigenschaften:

- DAX Queries dienen zur Abfrage einer Ergebnistabelle aus dem Datenmodell
- DAX Queries sind aufgrund des **EVALUATE** Statements im Syntax leicht zu erkennen und sehen häufig „SQL ähnlich“ aus

Automatismen/Wizards:

- **Performance Analyzer**






```
// DAX Query
DEFINE VAR __H0FilterTable =
  TREATAS({"A262"}, 'Dim Products'[Article Key])

EVALUATE
  SAMPLE(
    3502,
    SUMMARIZECOLUMNS(
      'Fact Sales'[Order Date],
      "Sales", ' Measures'[Sales],
      "v_MeasuresSales", IGNORE(CALCULATE(' Measures'[Sales], KEEPFILTERS(__H0FilterTable)))
    ),
    'Fact Sales'[Order Date],
    1
  )

ORDER BY
  'Fact Sales'[Order Date]
```

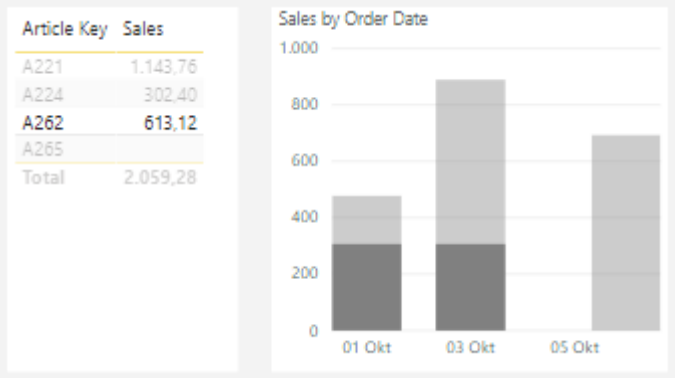
Power BI Camp DAX mit Power BI

Automatismen/Wizards:

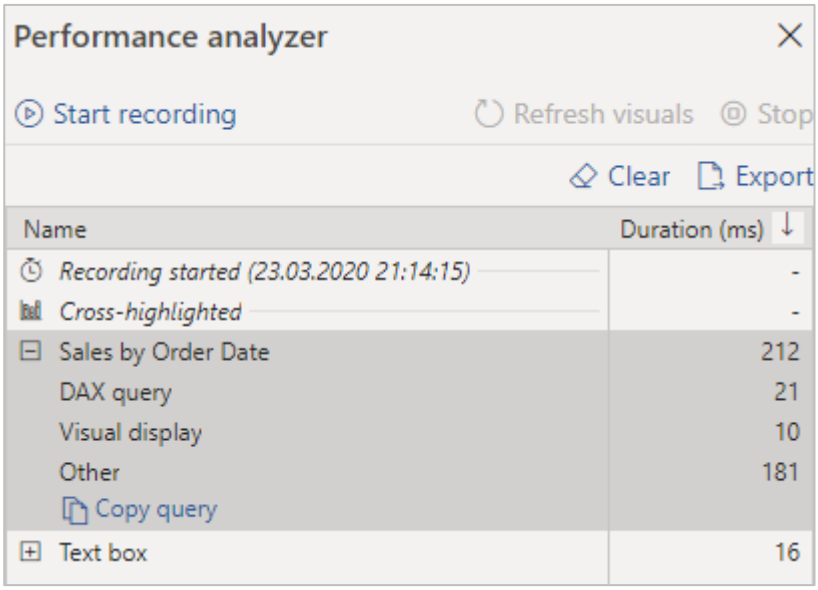
- Performance Analyzer

  Einen guten ersten Eindruck zu DAX Queries kann man sich mit der Funktion **Performance Analyzer** (in Power BI Desktop) verschaffen. Das Tool protokolliert neben der Laufzeit aller ausgeführten Prozesschritte auch die DAX Queries, die von den Visuals an das Datamodel abgesetzt werden (ja: die Visuals kommunizieren Power BI intern mittels DAX Queries mit dem Datamodel).

  Hier ein sehr einfacher Report, bei dem im linken Table Visual ein Cross Filter auf einen Artikel gesetzt wird und damit das Chart Visual rechts gefiltert wird:



  Der Performance Analyzer protokolliert dabei folgende Steps (samt Laufzeiten in Millisekunden) und macht auch die DAX Query mit der Funktion **Copy query** zugänglich:



Typisch für ein DAX Query ist das Codewort EVALUATE, mit dem jede DAX Query eingeleitet wird:

```
// DAX Query
DEFINE VAR __H0FilterTable =
  TREATAS({"A262"}, 'Dim Products'[Article Key])

EVALUATE
  SAMPLE(
    3502,
    SUMMARIZECOLUMNS(
      'Fact Sales'[Order Date],
      "Sales", ' Measures'[Sales],
      "v_MeasuresSales", IGNORE(CALCULATE(' Measures'[Sales], KEEPFILTERS(__H0FilterTable)))
    ),
    'Fact Sales'[Order Date],
    1
  )

ORDER BY
  'Fact Sales'[Order Date]
```

3.2.6 ABGRENZUNGEN

Was es in DAX nicht gibt:

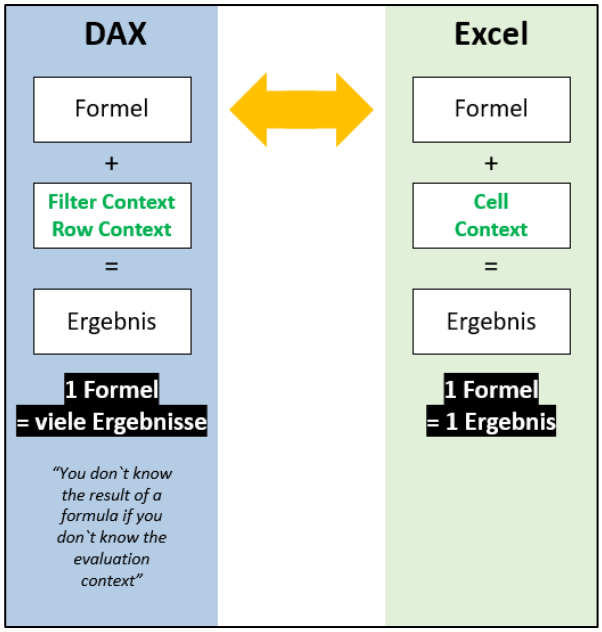
- Keine Calculated Rows innerhalb einer bestehenden Tabelle
  Es können aber mittels Append-Query einer bestehenden Tabelle Datensätze hinzugefügt werden.

- Keine Calculated Elements innerhalb einer importierten Column (wie in MDX Cubes oder im Excel Pivot Cache)
  Es kann jedoch ein Measure so definiert werden, daß auf importierten (Träger-)Elementen ausgewählte Berechnungen ausgeführt werden und damit die Funktionalität von Calculated Elements nachempfunden wird

3.3 DAX EVALUATION CONTEXT („BAUSTEIN 2")

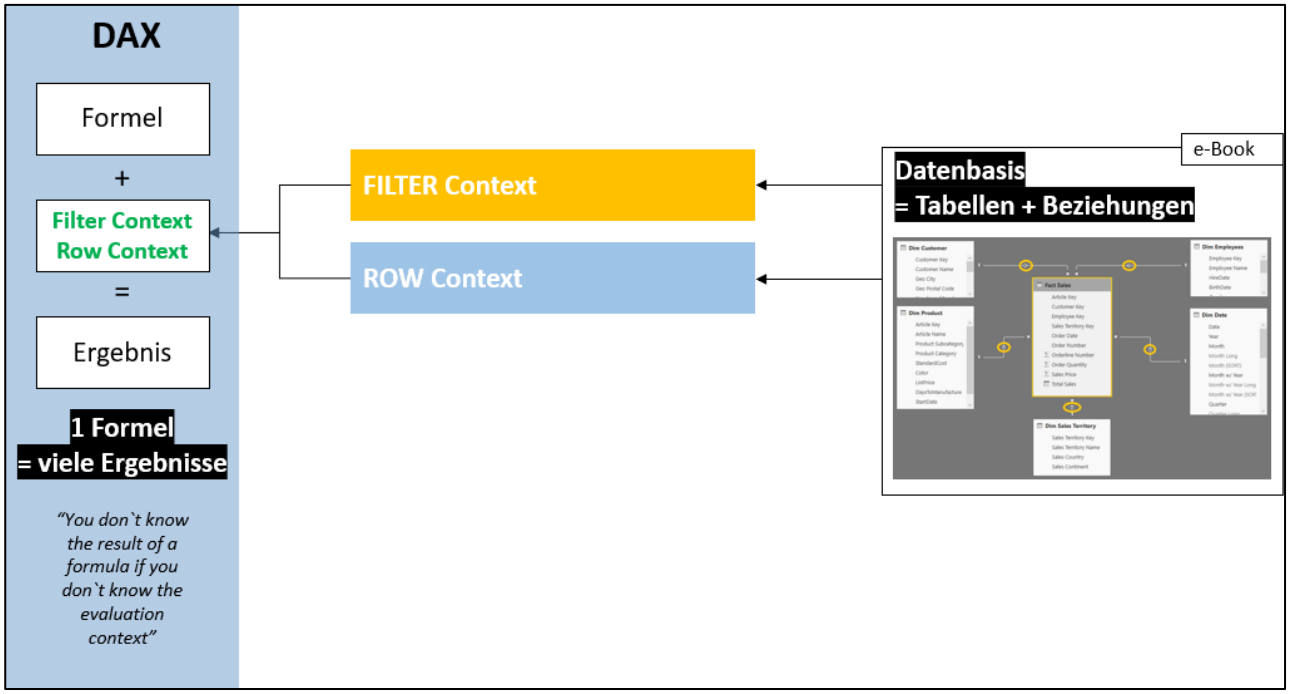
3.3.1 DAX FORMULA EVALUATION

Evaluation bedeutet im Zusammenhang mit DAX die konkrete Ermittlung eines Ergebnisses aus einer DAX Formel. Das Ergebnis einer Formel ist nur in Verbindung mit dem Context – vergleichbar wie in Excel – ableitbar:



Natürlich hängt das Ergebnis einer Berechnung primär vom importierten Datenmaterial in den Tabellen des Datamodels ab (wie in Excel das Ergebnis einer Formel vom Inhalt der referenzierten Zellen abhängt). DAX-Berechnungen in Measures und Calculated Columns beruhen auf 2 spezifischen (grundsätzlich sehr einfachen) Konzepten:

1. Filter Context
2. Row Context

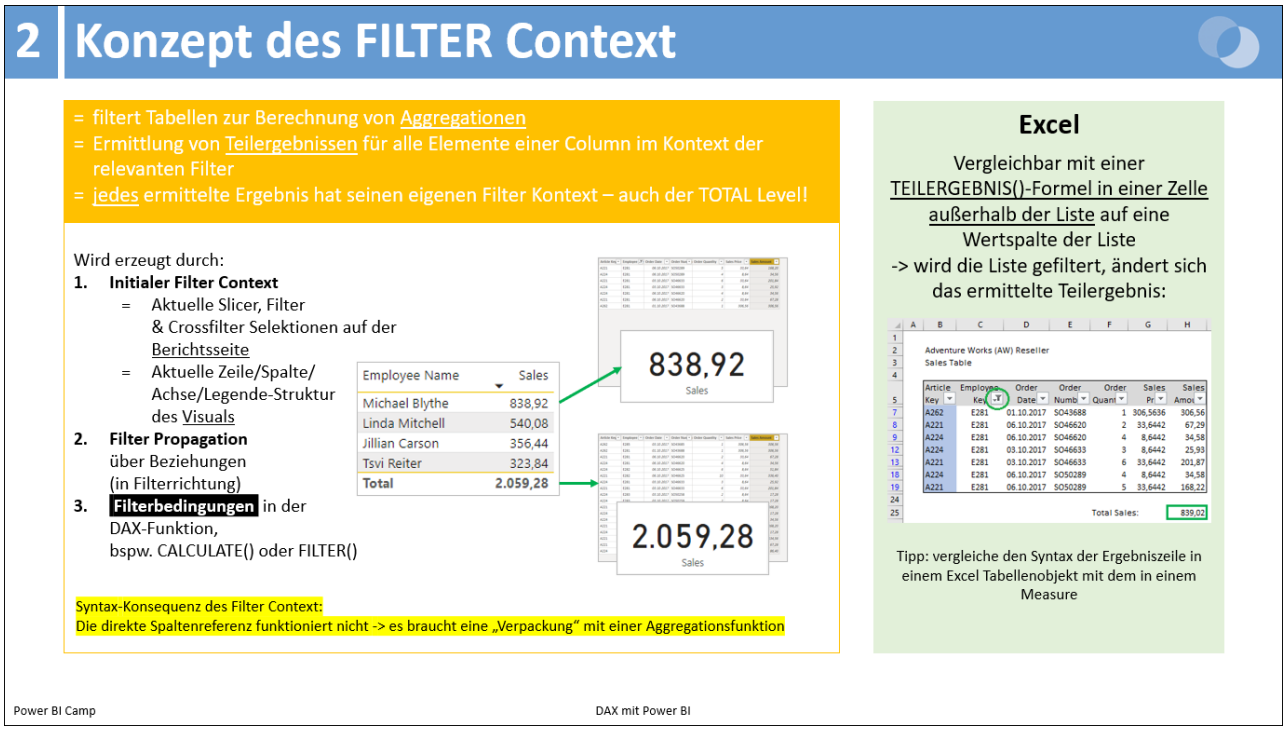


Zur Berechnung eines konkreten Ergebnisses werden folgende Evaluierungsschritte angewendet:

1. Filter auf die jeweilige (Dimensions-)Tabelle anwenden
   - "Initialen (impliziten) Filter Context" aus dem Visual / dem PivotTable ermitteln
   - "Explizite Filter" ermitteln (falls vorhanden, bspw. aus einem CALCULATE()-Statement) und übersteuern der impliziten Filter
2. Beziehungen anwenden = "Filter Propagation" über die verknüpften Keys in die definierte Filterrichtung
3. Berechnen der Expression (Formel) auf die aus Schritt 2 gefilterte (Fakten-)Tabelle = Filter Context (bei Verwendung einer Iterator Funktion im angegebenen Row Context)

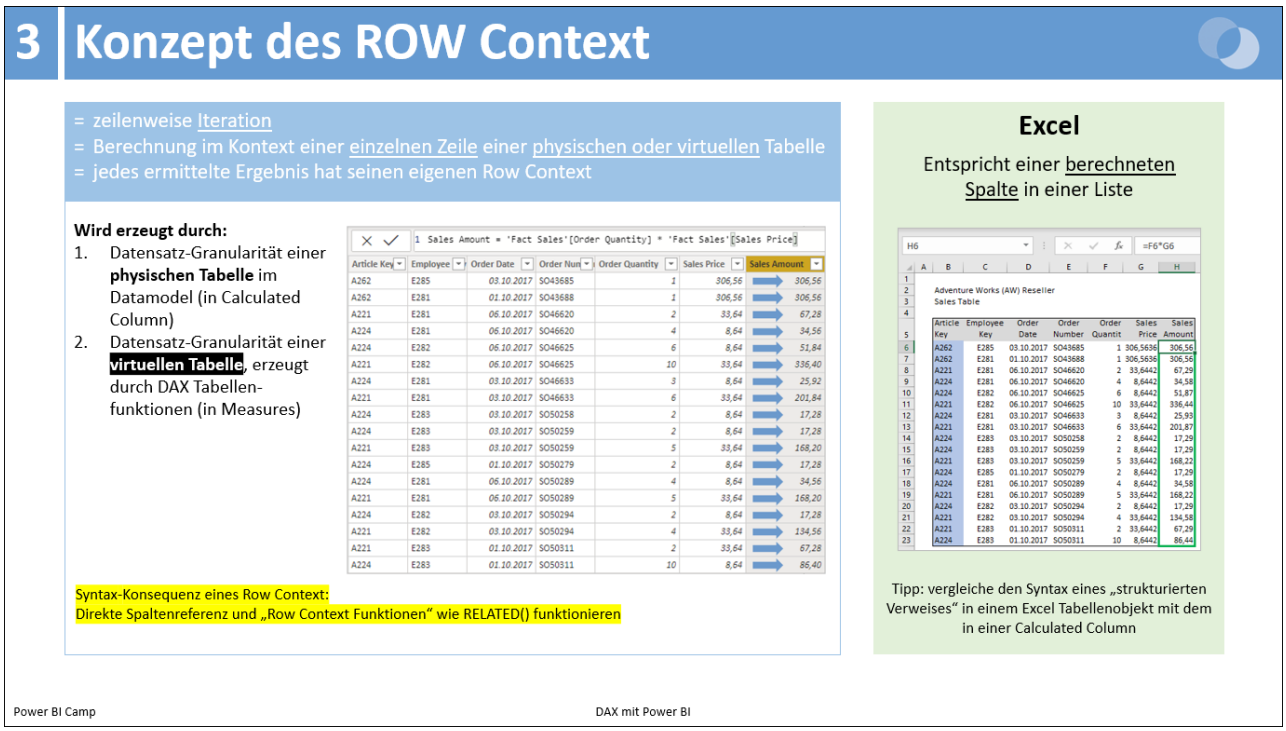
Weiterführend:
https://www.powerbi-pro.com/calculate-das-schweizer-armeemesser-in-dax/

3.3.2 KONZEPT DES FILTER CONTEXT



Konkret heißt das, daß das Measure „Sales" im Visual links 5 x in 5 verschiedenen Filter Contexts berechnet wird und damit 4 gefilterte Teilergebnisse und 1 ungefiltertes Gesamtergebnis ermittelt wird.

3.3.3 KONZEPT DES ROW CONTEXT



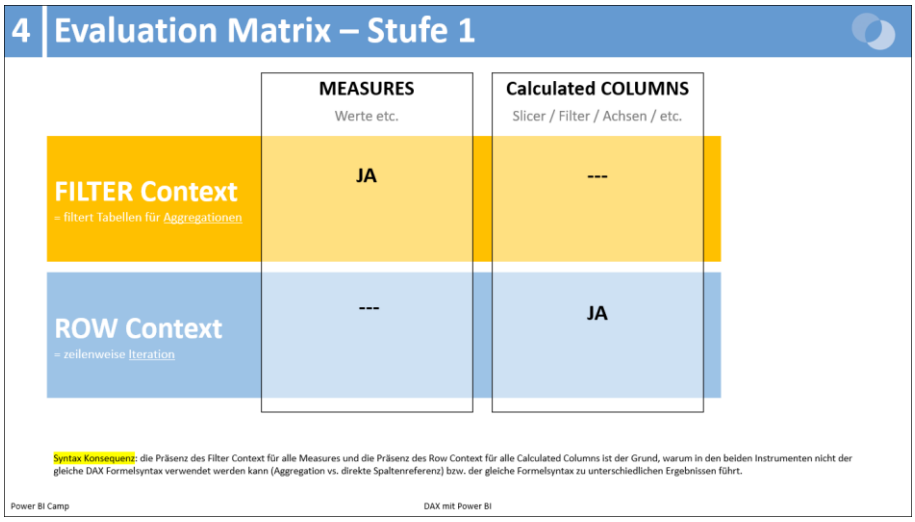
Konkret heißt das, daß die Calculated Column „Sales Amount" in der Tabelle unten 18 x in 18 verschiedenen Row Contexts berechnet wird und damit 18 spezifische Berechnungsergebnisse ermittelt werden (etwas anderes würden wir auch nicht erwarten).

Tipp: vergleiche den Syntax eines „strukturierten Verweises" in einem Excel Tabellenobjekt mit dem in einer Calculated Column – diese sind fast ident!

3.3.4 EVALUATION MATRIX – STUFE 1 UND 2

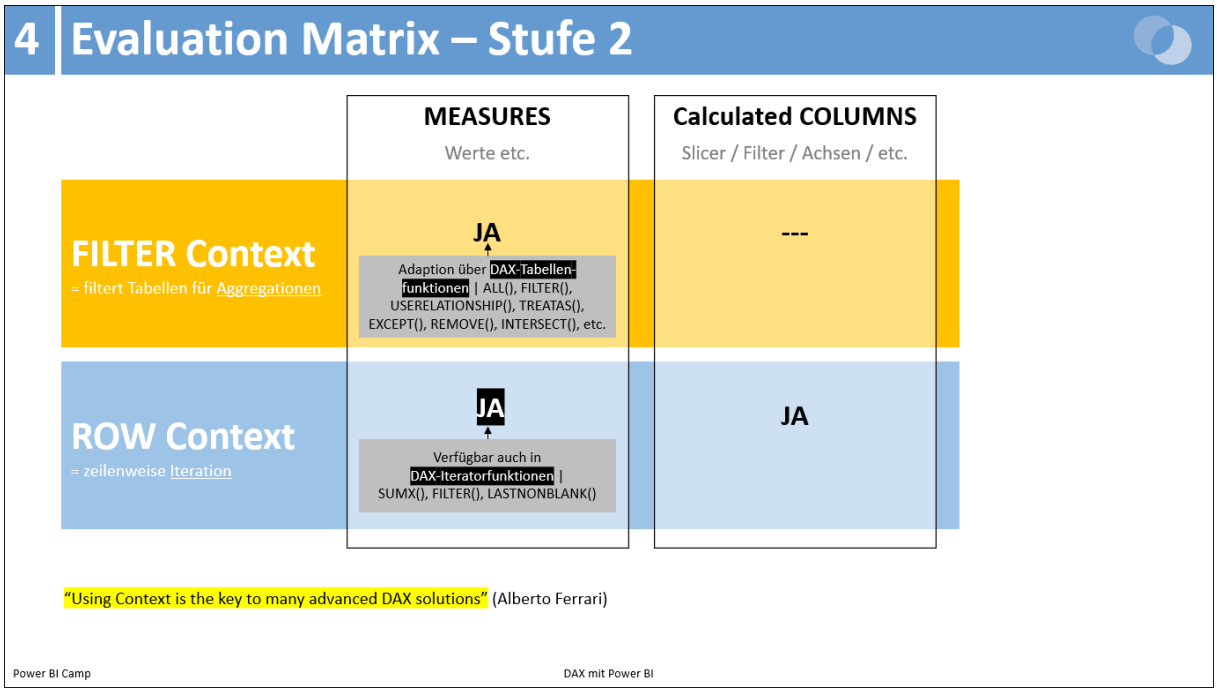
Grundsätzlich ist alles sehr einfach:

- **Measures** arbeiten immer in einem Filter Context und **Calculated Columns** immer in einem Row Context.
- Das ist auch der Grund, warum in den beiden Instrumenten idR nicht exakt der gleiche DAX Formelsyntax verwendet werden kann (Aggregation vs. direkte Spaltenreferenz) bzw. der gleiche Formelsyntax zu unterschiedlichen Ergebnissen führt.



Wäre das alles, wären die Berechnungsmöglichkeiten in DAX sehr überschaubar.

- Mit den DAX-Tabellenfunktionen – iVm einem CALCULATE() oder SUMX() – kann der Filter Context adaptiert werden und damit sehr mächtige Berechnungen wie Year-to-Date, Year-over-Year, %-von-Gesamt und vieles mehr bewerkstelligt werden.
- Mit den DAX-Iteratorfunktionen – bspw. SUMX(), AVERAGEX(), RANKX(), FILTER() u.a. – können zweistufige Berechnungen zuerst in einem virtuellen Row Context und dann in einem Filter Context ausgeführt werden und damit bspw. die jeweils höchsten Tagesumsätze eines Monats im Berichtsjahr ermittelt werden.



3.3.5 KONZEPT DER CONTEXT TRANSITION

5 Konzept der Context Transition

Konzept
- **transformiert den Row Context – bspw. die aktuelle Artikelnummer – in einen äquivalenten Filter Context für eine Berechnung**
- **kann verwendet werden in einem Row Context, d.h. in einer Calculated Column oder einer Iteratorfunktion**

Context Transition wirkt, wenn
- ein **Measure** verwendet wird (anstelle einer Aggregationsfunktion)
- ein **CALCULATE**() oder **CALCULATETABLE**() als „Wrapper" für eine Aggregationsfunktion verwendet wird
- ein **RELATEDTABLE**() verwendet wird (bspw. in Verbindung mit einem COUNTROWS)

Zur Verdeutlichung:
- Ermittlung Gesamterlös aus Faktentabelle als Calculated Column in der Dimensionstabelle „Dim Product"

DAX Patterns:
- C06.2 Lookup in verknüpfter Faktentabelle
- C07.2 Bedingter Lookup für Provisionssätze, Steuersätze usw.
- C07.3 Bedingter Lookup für Währungsumrechnung
- C07.4 Bedingter Lookup für Slowly-Changing-Dimensions
- C07.5 Bedingter Lookup für Statische Segmentierung nach Preisklassen
- C08 Statische Segmentierung nach Neukunden
- C09 Statische Segmentierung nach Kleinkunden
- C10 Erster Timestamp innerhalb eines Process-Keys (als Column)

Article Key	Article Name	Product Cat	Product Sub	Standard Cos	List Price	CT Sales 1	CT Sales 2
A221	Sport-100 Helmet, Blue	Accessories	Helmets	13,88 €	33,64 €	2.059,28	1.143,76
A224	AWC Logo Cap	Clothing	Caps	5,23 €	8,64 €	2.059,28	302,40
A262	LL Road Frame - Red, 44	Components	Road Frames	181,49 €	306,56 €	2.059,28	613,12
A265	LL Road Frame - Red, 48	Components	Road Frames	187,16 €	337,22 €	2.059,28	

```
CT Sales 1 = SUM('Fact Sales'[Sales Amount])
```

```
CT Sales 2 =
  CALCULATE(
    SUM('Fact Sales'[Sales Amount])
  )
```

Power BI Camp DAX mit Power BI

5 Konzept der Context Transition

Weitere Beispiele:
- Ermittlung, ob ein Dimensionselement Einträge in der Faktentabelle hat oder nicht

Article Key	Article Name	Product Cat	Product Sub	Standard Cos	List Price	CT Records 1	CT Records 2	CT Has Records
A221	Sport-100 Helmet, Blue	Accessories	Helmets	13,88 €	33,64 €	18	7	Yes
A224	AWC Logo Cap	Clothing	Caps	5,23 €	8,64 €	18	9	Yes
A262	LL Road Frame - Red, 44	Components	Road Frames	181,49 €	306,56 €	18	2	Yes
A265	LL Road Frame - Red, 48	Components	Road Frames	187,16 €	337,22 €	18		No

```
CT Records 1 = COUNTROWS('Fact Sales')
```

```
CT Records 2 =
  COUNTROWS(
    RELATEDTABLE('Fact Sales')
  )
```

```
CT Has Records =
  IF(ISBLANK(
    COUNTROWS(
      RELATEDTABLE('Fact Sales')
    ));
    "No":"Yes"
  )
```

- (Lästige) Syntaxanforderung für Measures mit Iteratorfunktionen

```
Max Daily Sales =
  MAXX(
    VALUES('Dim Dates'[Date]);
    SUM('Fact Sales'[Sales Amount])
  )
```

```
Max Daily Sales =
  MAXX(
    VALUES('Dim Dates'[Date]);
    CALCULATE(SUM('Fact Sales'[Sales Amount]))
  )
```

```
Max Daily Sales =
  MAXX(
    VALUES('Dim Dates'[Date]);
    [Sales]
  )
```

Power BI Camp DAX mit Power BI

Weiterführend:
https://www.sqlbi.com/articles/understanding-context-transition/

3.3.6 KONZEPT DES OUTER FILTER CONTEXT

6 Konzept des Outer Filter Context

Konzept:
- **Variablen** erzeugen bei der Ermittlung eines Skalarwertes eine Konstante
- Variablen werden damit dort berechnet, wo sie definiert sind und nicht wo sie verwendet werden

Ein **Outer Filter Context** ist verfügbar, wenn
- Eine Variable bspw. außerhalb des CALCULCATE() oder SUMX() Funktion platziert wird

DAX Filterfunktionen:
- EARLIER()
- EARLIEST()

```
Quantity (TOTAL) =
    CALCULATE(
        [Quantity];
        ALL('Fact Sales')
    )
```

```
Quantity (TOTAL - funktioniert nicht) =
VAR
    varSales = [Quantity]
RETURN
    CALCULATE(
        varSales;
        ALL('Fact Sales')
    )
```

Power BI Camp DAX mit Power BI

Siehe auch Kapitel „Variablen in DAX".

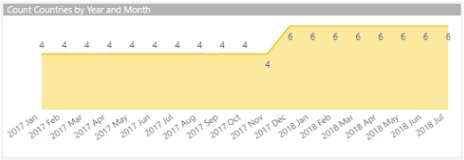
3.3.7 KONZEPT DER EXPANDED TABLES

7 Konzept der Expanded Tables

Konzept
- **systemintern werden alle Felder einer verknüpften Dimensionstabelle an die Faktentabelle angehängt** (exakt: alle Felder auf der 1-Seite einer 1:n Verbindung werden an die Tabelle auf der n-Seite angehängt.
- **daher berücksichtigt die Berechnung mittels Expanded Tables auch alle Filter, die auf die Faktentabelle wirken, auch in den Dimensionstabellen, obwohl es keine aktive Filterbeziehung dorthin gibt** (als wäre eine bidirektionale Beziehung aktiv).
- ist nicht verfügbar bei „Weak Relations" (m:n Beziehung, Cross-Island Beziehung in Composite Models)
- die Berechnung ist jedoch aufwendig und daher langsam

Ein **Expanded Table** ist verfügbar, wenn
- ein Tabellenname als Filterbedingung in einem **CALCULATE**() verwendet wird

Zur Verdeutlichung:
- Ermittlung eines DISTINCTCOUNT() in einer Dimensionstabelle (bspw. Anzahl der Länder, der Produktgruppen, der Monate, usw.)




```
Count Countries =
  CALCULATE(
    DISTINCTCOUNT('Dim Customers'[Geo Country]);
    'Fact Sales'
  )
```

DAX Patterns:
- M04.2 Anzahl Länder, Produktgruppen, Monate, usw. = Ermittlung Distinctcount in der Dimensionstabelle
- M07.8 Moving-12-Month-Average

Alternative:

```
Count Countries (ohne ET) =
  CALCULATE(
    DISTINCTCOUNT('Dim Customers'[Geo Country]);
    CROSSFILTER(
      'Fact Sales'[Customer Key];
      'Dim Customers'[Customer Key];
      Both
    )
  )
```

Power BI Camp DAX mit Power BI

Zur Verdeutlichung:
- Ermittlung eines DISTINCTCOUNT() in einer Dimensionstabelle (bspw. Anzahl der Länder, der Produktgruppen, der Monate, usw.)

  Measure:

```
Count Countries (mit ET) =
  CALCULATE(
    DISTINCTCOUNT('Dim Customers'[Geo Country]);
    'Fact Sales'
  )
```

```
Count Countries (funktioniert nicht) =
  DISTINCTCOUNT('Dim Customers'[Geo Country])
```

Die im Filterbereich angegebene Tabelle „Fact Sales" ist nicht nur die Tabelle selbst sondern der Expanded Table, d.h. alle Felder der Dimensionstabellen sind (wie in einem Flat Table) angehängt. Daher berücksichtigt die Berechnung mittels Expanded Tables auch alle Filter, die auf die Faktentabelle wirken, auch in den Dimensionstabellen, obwohl es keine aktive Filterbeziehung dorthin gibt.

Date	Count Countries (mit ET)	Count Countries (funktioniert nicht)
01.01.2017	4	6
02.01.2017	1	6
03.01.2017	2	6
04.01.2017	2	6
05.01.2017	1	6
06.01.2017	3	6
07.01.2017	4	6
08.01.2017	4	6
09.01.2017	1	6
10.01.2017	2	6
11.01.2017	3	6
Total	**6**	**6**

Die Berechnung im zweiten Measure funktioniert nicht, da nur im Customer Table gezählt wird und der Filter Context nur aus der eigenen Tabelle kommt, nicht aber aus den anderen Dimensionstabellen.

Alternativen zur Berechnung:

- Der DISTINCTCOUNT() auf das Dimensionsattribut würde ohne Expanded Tables ebenfalls funktionieren, wenn entweder die Beziehung zwischen Faktentabelle und Kundendimension auf bidirektional umgestellt wird (und ein Filter aktiv) ist oder die CROSSFILTER()-Funktion verwendet wird:

  Measure:

```
Count Countries (mit Crossfilter) =
   CALCULATE(
      DISTINCTCOUNT('Dim Customers'[Geo Country]);
      CROSSFILTER(
         'Fact Sales'[Customer Key];
         'Dim Customers'[Customer Key];
         Both
      )
   )
```

  Ob die Berechnung mit dem CROSSFILTER() tatsächlich schneller ist als mit dem Expanded Table können wir bisher aus eigener Erfahrung nicht bestätigen.

- Das Dimensionsattribut könnte mit der RELATED()-Funktion in einer Calculated Column in die Faktentabelle geholt werden und dann ein „einfacher" DISTINCTCOUNT auf die Faktentabelle ausgeführt werden (ist aber natürlich unflexibel und verkompliziert das physische Datenmodell).

Weiterführend:

- https://www.sqlbi.com/articles/expanded-tables-in-dax/
- https://www.sqlbi.com/articles/using-the-selectedvalue-function-in-dax/ (auch hier geht es um Expanded Tables)
- https://blog.enterprisedna.co/2017/11/27/counting-customers-over-time-dax-example-in-power-bi
- https://docs.microsoft.com/en-us/power-bi/desktop-relationships-understand#strong-relationships:

  „At query time, strong relationships permit table expansion to take place. Table expansion results in the creation of a virtual table by including the native columns of the base table and then expanding into related tables. For Import tables this is done in the query engine; for DirectQuery tables it is done in the native query sent to the source database (providing that the "Assume referential integrity" property isn't enabled). The query engine then acts upon the expanded table, applying filters and grouping by the values in the expanded table columns."

  "Inactive relationships are expanded also, even when the relationship isn't used by a calculation. Bi-directional relationships have no impact on table expansion."

  "For One-to-many relationships, table expansion takes place from the "many" to the "one" sides by using LEFT OUTER JOIN semantics. Table expansion also occurs for One-to-one intra-island relationships, but by using FULL OUTER JOIN semantics. It ensures that blank virtual rows are added on either side, when necessary. The blank virtual rows are effectively Unknown Members. Unknown members represent referential integrity violations where the "many" side value has no corresponding "one" side value. Ideally these blanks should not exist, and they can be eliminated by cleansing or repairing the source data."

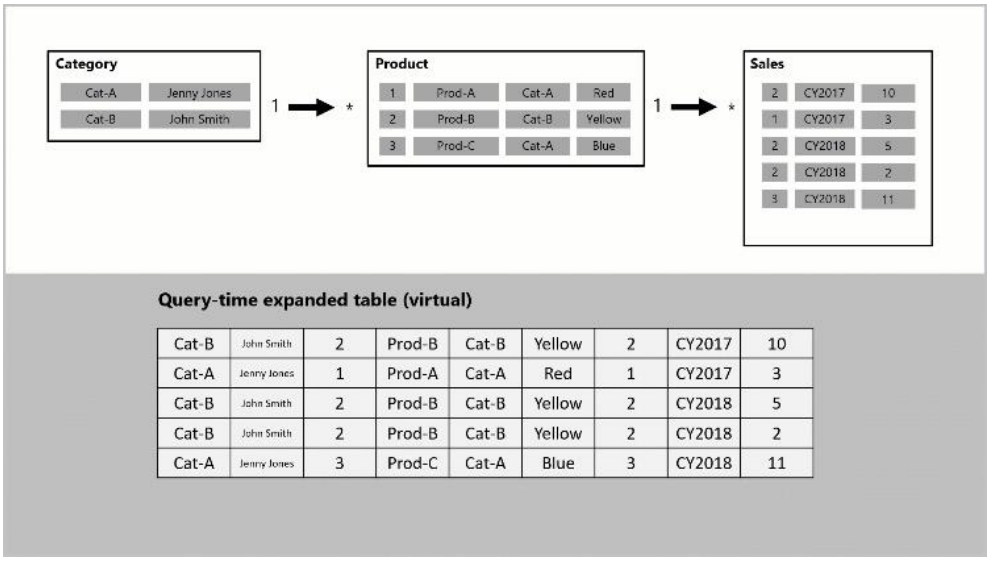
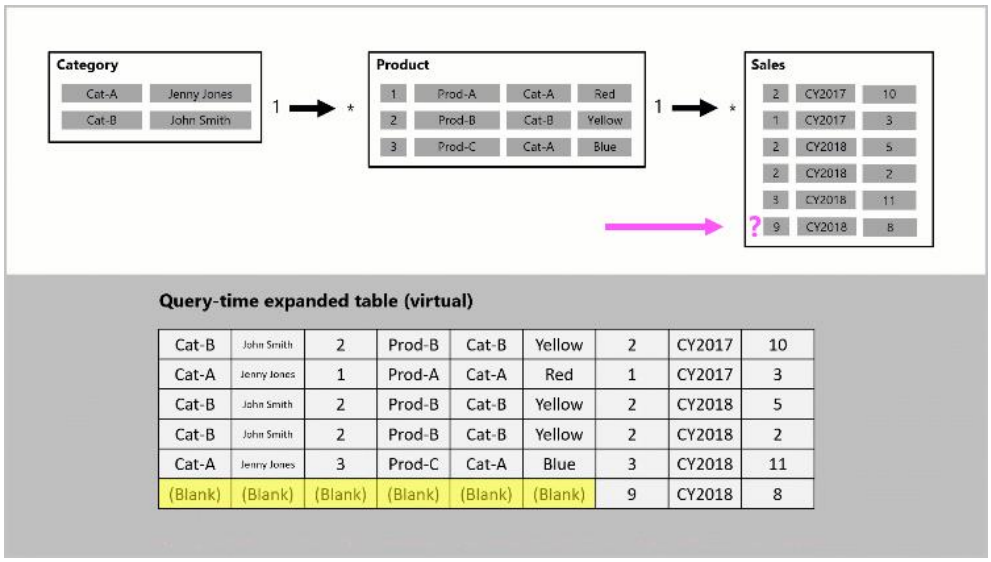


Abbildung: Expanded Table "Sales" ohne Blankrow.



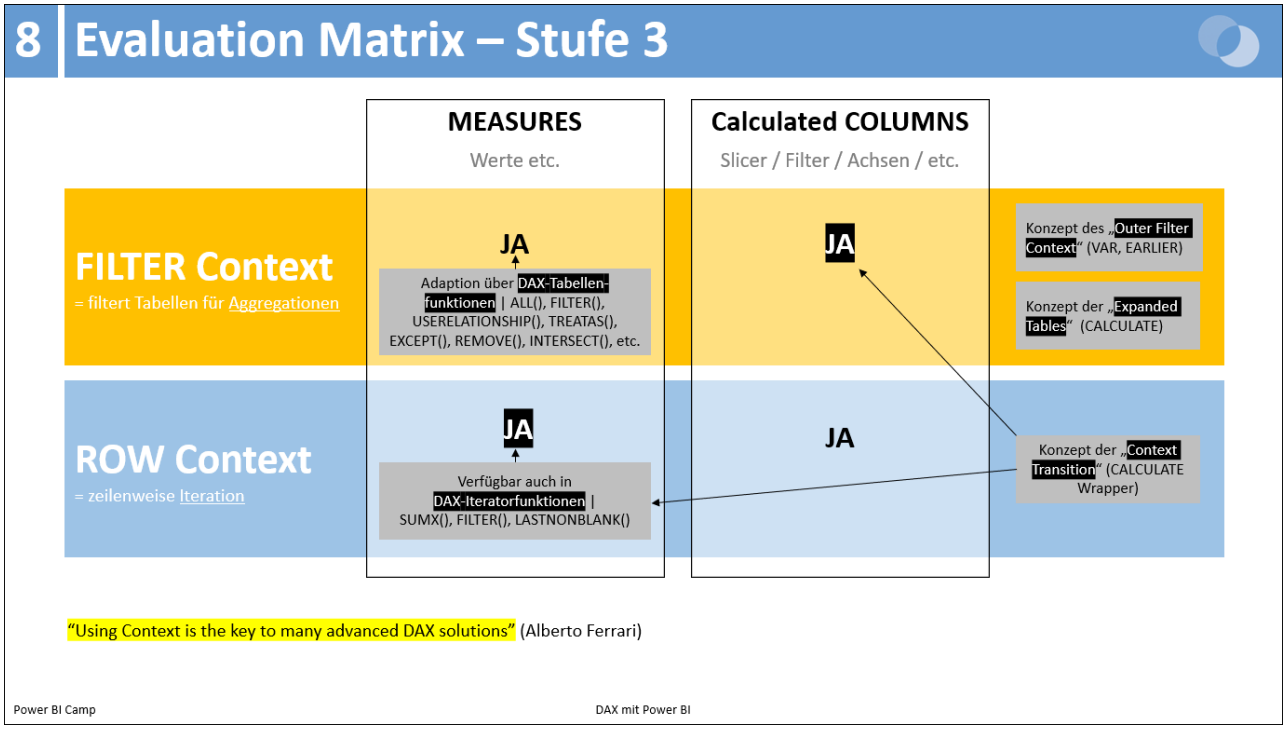
Cat-B	John Smith	2	Prod-B	Cat-B	Yellow	2	CY2017	10
Cat-A	Jenny Jones	1	Prod-A	Cat-A	Red	1	CY2017	3
Cat-B	John Smith	2	Prod-B	Cat-B	Yellow	2	CY2018	5
Cat-B	John Smith	2	Prod-B	Cat-B	Yellow	2	CY2018	2
Cat-A	Jenny Jones	3	Prod-C	Cat-A	Blue	3	CY2018	11
(Blank)	(Blank)	(Blank)	(Blank)	(Blank)	(Blank)	9	CY2018	8

Abbildung: Expanded Table "Sales" mit Blankrow.

3.3.8 EVALUATION MATRIX – STUFE 3

Es geht aber noch weiter:

- Das Konzept der Context Transition ermöglicht die Übergabe des Row Context der aktuellen Zeile (bspw. in einer Calculated Column) als Filterbedingung für eine Berechnung. Damit lassen sich zahlreiche praxisrelevante Patterns abbilden.
- Das Konzept des Outer Filter Context ermöglicht weitere Adaptionsmöglichkeiten des Filter Context.
- Das Konzept der Expanded Tables ermöglicht Berechnungen auf die verknüpften Dimensionstabellen auf Basis der Faktentabelle.



3.4 DAX SYNTAX ("BAUSTEIN 3")

3.4.1 ÜBERSICHT

DAX Formeln		Excel Formeln
1. Name = DAX Statement	Umsatz = SUM([Tabelle][Spalte])	Zelle * = SUM(E1:E6)
2. Funktionen	SUM(), FILTER(), CALCULATE(), u.v.m.	Ja einige idente Funktionen
3. Referenzen	zu Tabellen, Spalten und Measures	Ja aber zu Zellen und benannten Bereichen
4. Operatoren	= == < > + - * / && AND || OR u.a.	Ja aber keine Spezial-Operatoren
5. Konstanten	=5 =„Text" =DATE() =BLANK()	Ja aber Leerstring "" statt BLANK()
6. Datentypen	Zahl, Text, Datum, BLANK u.a.	Ja aber es handelt sich um Zellformate
7. Kommentare	// und /* … */	Nein
8. Formatierung	Only formatted DAX is DAX!	Ja aber selten verwendet
9. Variablen	VAR … RETURN	Nein
10. Skalare	Tabelle mit 1 Spalte und 1 Zeile	Ja jede Zelle liefert einen Skalarwert
11. Syntax Sugar	SUM() = SUMX() u.v.m.	Ja aber weniger stark ausgeprägt

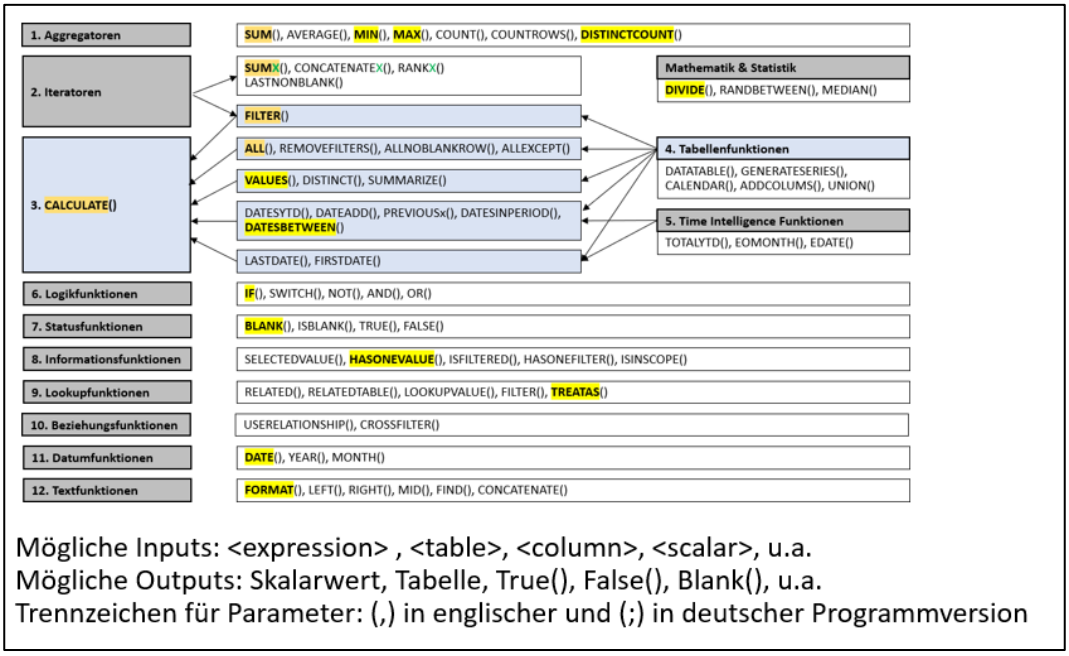
3.4.2 ALLGEMEINE DAX SYNTAX REGELN

- Name = DAX Statement

Sales = SUM([Tabelle][Spalte])

- Ungültige Zeichen für Objektnamen: .,;':/*|?&$!+=()[]{}<>
- In Tabular Model und Power Pivot muss für Measures „:=" statt „=" verwendet werden, Calculated Columns können erst nach der Anlage auf den gewünschten Namen umbenannt werden.
- Tipp: auch (gültige) Sonderzeichen in Measure Namen verwenden -> benutzerfreundliche Bezeichnungen
  Profit_in_Proz
  Profit in Proz.
  Profit in %

- Funktionen



Die zentralen Funktionen sind SUM(), SUMX() CALCULATE(), FILTER() und ALL(). Die Darstellung finden Sie im Kapitel „DAX Funktionen" in gut lesbarer Form!

Weiterführend:
https://msdn.microsoft.com/en-us/query-bi/dax/dax-function-reference

- Referenzen

Referenzen zu Tabellen, Spalten und Measures:

- 'Table Name'
  Tabellennamen mit Leerzeichen in einfache Anführungszeichen

- 'Table Name'[Column Name]
  Spaltennamen in eckigen Klammer
  Referenzen zu Columns sollten immer den Tabellennamen enthalten, auch in Calculated Columns (obwohl das dort technisch nicht notwendig ist) -> bessere Lesbarkeit

- [Measure Name]
  Measure-Namen in eckigen Klammern
  Referenzen zu Measures sollten niemals den Tabellennamen enthalten
  'Table Name'[Measure Name] -> unnötig und Fehler bei Änderung des "Home Tables"
  > das Statement ist einfacher zu lesen
  > das Measure kann unkompliziert in eine andere Tabelle umgehängt werden
  > die sog. **Context Transition** für Measures (mit einem unsichtbaren CALCULATE-Wrapper) ist leichter zu verstehen bzw. im Code zu sehen

Bei Updates des Datenmodells werden die Referenzen automatisch aktualisiert.

- Operatoren

  - Arithmetik: +, -, *, /, ^
  - Klammern: ()
    für Vorrangregeln und/oder Gruppierung von Argumenten
  - Vergleich: =, ==, >=, >, <=, <
    <> = NOT()
    Der Vergleich ist nur zwischen 2 numerischen oder zwischen 2 nicht-numerischen **Datentypen** möglich.
  - Logik: A && B = AND (A, B)
    A || B = OR (A, B) = IN { "A", "B" }
    AND und OR -> nur 2 Bedingungen
    && and || -> viele Bedingungen
  - Textverkettung: &

  DAX Logikfunktionen:
  - AND()
  - OR(), IN{}
  - NOT()
  - CONTAINS()

  Textverkettung
  - CONCATENATE()
  - CONCATENATEX()

Alle Vergleichsoperatoren außer == behandeln BLANK als identisch zur Zahl 0, leere Zeichenfolge "", DATE(1899, 12, 30) oder FALSE. Infolgedessen ist [Spalte] = 0 true, wenn der Wert von [Spalte] entweder 0 oder BLANK ist. Im Gegensatz dazu ist [Spalte] == 0 nur true, wenn der Wert von [Spalte] 0 ist.

Weiterführend:
https://docs.microsoft.com/de-de/dax/dax-operator-reference
https://www.sqlbi.com/articles/the-in-operator-in-dax/

- Konstanten

  Beispiele:
  - Zahl = 5
  - Text = "Deutschland" oder "5" oder ""
    = DATEVALUE("31.05.2020")
  - Datum = DATE(2020;5;31)
  - Leer = BLANK()

  Excel:
  = 5
  = Deutschland oder ,5
  = 31.05.2020
  = 31.05.2020
  = ""

  Der **Datentyp** ist ausschlaggebend für die Vergleichsfähigkeit!

  Beispiel: DATE() kann in CALCULATE() als Filterparameter verwendet werden, DATEVALUE() aufgrund der Texteigenschaft hingegen nicht.

- Kommentare

  - Einzeilige Kommentare: //
  - Mehrzeilige Kommentare: /* … */

```
1 Sales YTD =
2     TOTALYTD(
3         [Sales];
4         'Dim Dates'[Date Key]
5     )
6 // This is a year-to-date calculation
```

- Formatierung

Only formatted DAX is DAX!

- Zeilenumbruch: [ALT] + [Enter] oder [Shift] + [Enter]
- Zeileneinrückung: [Tab] oder [Shift] + [Tab] oder [Backspace]

```
1 Sales YTD =
2     TOTALYTD(
3         [Sales];
4         'Dim Dates'[Date Key]
5     )
```

Weiterführend:
https://msdn.microsoft.com/en-us/query-bi/dax/dax-syntax-reference

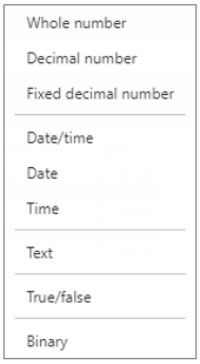
3.4.3 DATENTYPEN IN DAX

DAX kennt 8 Datentypen, vereinfacht gesagt sind das **Zahl, Datum und Text**. Die Datentypen sind wichtig, weil Funktionen nur bestimmte Datentypen als Parameter akzeptieren und weil Vergleiche nur innerhalb des gleichen Datentyps funktionieren.

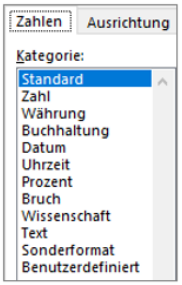
a. Numerisch

1. Integer (= Ganzzahl)
2. Decimal (= Dezimalzahl)
3. Fixed Decimal (= Feste Dezimalzahl)
4. Date / DateTime / Time (= Datum, Uhrzeit)
   - Zahl vor dem Komma:
     Anzahl der Tage vor/nach dem 30.12.1899 (=0)
     Datum +1 = morgen, Datum -1 = gestern
   - Zahl nach dem Komma:
     Sekunden (1 / 24 * 60 * 60)
5. TRUE/FALSE (true =1, false = 0)

PBI Datamodel:



Excel Zellformate:



b. Nicht-Numerisch

7. Text (= unicode string, case sensitive)
8. Binary (= visualiserbar, aber nicht mit DAX auswertbar)

Excel arbeitet mit Zellformaten, Berechnungsunterschiede zu Power BI auch bei Datum.

Importierte Columns haben einen **"expliziten Datentyp"**, der in der Query festgelegt wird. Der Datentyp sollte keinesfalls im Datamodel übersteuert werden, da dies zu einem intransparenten Modell und zu möglichem Informationsverlust führt.

Measures und Calculated Columns haben einen **"impliziten Datentyp"**, der sich aus dem Berechnungsergebnis des DAX Statements ableitet. Darüber hinaus wirken folgende Konzepte:

- **Operator Overloading** = der Datentyp des Berechnungsergebnisses wird mit dem verwendeten Operator bestimmt:
  "5" + "4" = 9 | 5 & 4 = "54".
  Manchmal ist das nützlich, manchmal kontraproduktiv …
- **DAX-Konvertierungsfunktionen** = der Datentyp des Ergebnisses wird mit der Funktion bestimmt

DAX Konvertierungs-funktionen:
- CONVERT()
- FORMAT()
- DATE()

DAX Informationsfunktionen sollten nicht verwendet werden, da deren Anwendung unnötig ist. Die Funktionen checken nicht den einzelnen Wert sondern ohnehin das Format der ganzen Column bzw. des Measures -> dieses kann leicht ohne Formel direkt im Datamodel definiert bzw. eingesehen werden
Ausnahme: ISBLANK() ist eine nützliche Funktion

DAX Informations-funktionen:
- ISBLANK()
- ISNUMBER()
- ISTEXT()
- ISNONTEXT()
- usw.

BLANK() kann vereinfachend als 9. Datentyp angesehen werden:

- BLANK() hat keine besonderen Eigenschaften sondern wird in DAX wie die Zahl 0 gehandhabt. Alle Vergleichsoperatoren außer == behandeln BLANK als identisch zur Zahl 0, leere Zeichenfolge "", DATE(1899, 12, 30) oder als FALSE.
  Infolgedessen ist „[Spalte] = 0" wahr, wenn der Wert von [Spalte] entweder 0 oder BLANK ist.
  Im Gegensatz dazu ist „[Spalte] == 0" nur dann wahr, wenn der Wert der [Spalte] 0 ist.
- BLANK Einträge in einer Spalte
  vs. BLANK Row in einer Dimension

DAX Informations-funktionen:
- BLANK()
- ISBLANK()

Weiterführend:
https://www.sqlbi.com/articles/how-to-handle-blank-in-dax-measures

3.4.4 VARIABLEN IN DAX

VAR + RETURN in Measures und Calculated Columns

- Je ein VAR Statement pro Variable
  Syntax: VAR <name> = <expression>
  <name> : keine Sonder- und Leerzeichen, keine Zahlen am Beginn, doppelter Underscore
  <expression> : kann Skalarwert (= Measure, Konstante, usw.) oder Tabelle (= Filter) liefern

- Danach ein RETURN … und weiter mit dem eigentlichen DAX Statement

```
Profit ACT in % = Divide(
    Sum('Fact Sales'[ACTUAL Sales]);
    Sum('Fact Sales'[ACTUAL Profit])
    )
```

```
Profit ACT in % =
    VAR
        var1 = Sum('Fact Sales'[ACTUAL Sales])
    VAR
        var2 = Sum('Fact Sales'[ACTUAL Profit])
    RETURN
        Divide (var1;var2)
```

- Üblicherweise steht das VAR-RETURN am Beginn eines DAX Statements, ist aber auch innerhalb möglich
- Derzeit gibt es keine globalen Variablen in DAX

Allgemeine Argumente für den Einsatz von Variablen:
- Bessere Lesbarkeit, da Bausteine zentral und nur 1x ermittelt werden
- Bessere Performance, da keine wiederholenden Berechnungen
- Einfacheres Debugging durch temporäre Ausgabe der einzelnen Variablen

Besondere Funktion von Variablen: die Ermittlung eines Skalarwertes erzeugt eine Konstante

DAX Filterfunktionen:
- EARLIER()
- EARLIEST()

1. **Das ermöglicht einen "Outer Filter Context" für Berechnungen**
   d.h. Variablen werden dort berechnet, wo sie definiert sind und nicht wo sie verwendet werden

```
1 Quantity (TOTAL) =
2     CALCULATE(
3         [Quantity];
4         ALL('Fact Sales')
5     )
```

```
1 Quantity (TOTAL - funktioniert nicht) =
2 VAR
3     varSales = [Quantity]
4 RETURN
5     CALCULATE(
6         varSales;
7         ALL('Fact Sales')
8     )
```

Hinweis: der Outer Filter Context ist ein Ersatz für die (sehr unintuitiven) Funktionen EARLIER() und EARLIEST().

2. **Das ermöglicht eine Unterbrechung der Berechnungskette und damit die Vermeidung von nicht unterstützten Funktionskombinationen**: bspw. darf in einem CALCULATE() kein Skalar als Filterbedingung verwendet werden, das ist nur innerhalb einer FILTER()-Funktion möglich (siehe dazu mehr im nächsten Punkt)

Funktioniert nicht:

```
Sales Running Total (v1 Scalar) =
    CALCULATE(
        [Sales];
        'Dim Dates'[Date] <= MAX('Dim Dates'[Date])
    )
```

A function 'MAX' has been used in a True/False expression that is used as a table filter expression. This is not allowed.

```
Sales Running Total (v2 Scalar) =
    CALCULATE(
        [Sales];
        'Dim Dates'[Date] <= [Highest Date (relative)]
    )
```

The value for 'Highest Date (relative)' cannot be determined. Either the column doesn't exist, or there is no current row for this column.

Funktioniert:

```
Sales Running Total (v3 Scalar) =
VAR
    var_MaxDatum = MAX('Dim Dates'[Date])
RETURN
    CALCULATE(
        [Sales];
        'Dim Dates'[Date] <= var_MaxDatum
    )
```

Funktioniert auch:

```
Sales Running Total (v4 Scalar) =
    CALCULATE(
        [Sales];
        FILTER(
            ALL('Dim Dates');
            'Dim Dates'[Date] <= MAX('Dim Dates'[Date])
        )
    )
```

Weiterführend:
https://docs.microsoft.com/de-de/dax/var-dax
https://docs.microsoft.com/en-us/power-bi/guidance/dax-variables (Beispiel)
https://www.sqlbi.com/articles/variables-in-dax/ (Beispiel)
https://blog.enterprisedna.co/using-variables-in-dax-a-detailed-example/ (Beispiel)
Schönes Beispiel: Es können auch Tabellenfunktionen/Filterbedingungen für CALCULATE in Variablen ermittelt werden

```
Total Export Sales =
VAR
    ExportSales = FILTER( ALL( Sales[Channel] ), Sales[Channel] = "Export" )
VAR
    Days = 365
VAR
    IncludedDates = FILTER( ALL(Dates[Date]), Dates[Date] < TODAY() && Dates[Date] >= TODAY() - Days )
RETURN
    CALCULATE( [Total Sales],
        ExportSales,
        IncludedDates )
```

Hinweis: in DAX Queries können ebenfalls Variablen definierten werden, allerdings mit anderen Statements:

- ein DEFINE Statement für jede Variable
- danach ein EVALUATE

3.4.5 SKALARE IN DAX

Ein Skalar ist eine Tabelle mit 1 Spalte und 1 Zeile, oder vereinfacht gesagt: ein einzelner Wert analog zum Ergebnis in einer Zelle in Excel.
DAX Funktionen können als Input Skalare verlangen oder selbst Skalare als Output generieren.

Skalare in DAX werden erzeugt durch:
1. ein Measure
2. eine Aggregatorfunktion wie SUM(), AVERAGE(), MIN(), MAX()
3. eine Tabellenfunktion wie VALUES(Spalte), sofern nur 1 Datensatz „getroffen" wird. Daher auch SELECTEDVALUE().

Verschiedene Funktionen akzeptieren in bestimmten Parametern nur Skalare …

```
1 Maximaler Wert aus 2 Spalten = MAX('Dim Product'[List Price];
```

MAX(ColumnNameOrScalar1; [Scalar2])
Returns the largest numeric value or largest string in a column, or the larger value between two scalar expressions. Ignores logical values.

… oder keine Skalare (leider kommt das Wort Skalar in der Fehlermeldung idR nicht vor):

```
1 Sales Running Total (v1 Scalar) =
2     CALCULATE(
3         [Sales];
4         'Dim Dates'[Date] <= MAX('Dim Dates'[Date])
5     )
```

A function 'MAX' has been used in a True/False expression that is used as a table filter expression. This is not allowed.

FIRSTDATE() und LASTDATE() sind Tabellenfunktionen und liefern als Ergebniswert eine Tabelle (mit 1 Spalte und 1 Zeile), daher können diese Funktionen unmittelbar als Filterbedingung in einer CALCULATE()-Funktion verwendet werden:

```
1 Stock 1 (LASTDATE) =
2     CALCULATE(
3         SUM('Fact Table 1 (with 0)'[Stock]);
4         LASTDATE('Fact Table 1 (with 0)'[Date])
5     )
```

MIN() und MAX() sind Aggregatoren und liefern als Ergebniswert einen Skalar, daher können diese Funktionen nicht unmittelbar als Filterbedingung in einer CALCULATE()-Funktion eingesetzt werden (sondern müssen in eine FILTER()-Funktion gesetzt werden):

```
1 Stock 1 (LASTDATE) =
2     CALCULATE(
3         SUM('Fact Table 1 (with 0)'[Stock]);
4         MAX('Fact Table 1 (with 0)'[Date])
5     )
```

A function 'MAX' has been used in a True/False expression that is used as a table filter expression. This is not allowed.

3.4.6 SYNTAX SUGAR

Zahlreiche DAX Funktionen dienen lediglich dazu, den ansonsten teilweise anspruchsvollen Syntax zu vereinfachen. Hier einige prominente Beispiele:

SUM() = ein gekürztes SUMX()

```
Summe (Syntax Sugar 1) =
    SUM('Fact Sales'[Sales Amount])
```

```
Summe (Syntax Sugar 2) =
    SUMX(
        'Fact Sales';
        'Fact Sales'[Sales Amount]
    )
```

MIN()/MAX() mit 2 Parameter = gekürzter IF-Vergleich

```
MIN-MAX Comparison (Syntax Sugar 1) =
    MIN(
        AVERAGE('Dim Products'[List Price]);
        AVERAGE('Dim Products'[Standard Cost])
    )
```

```
MIN-MAX Comparison (Syntax Sugar 2) =
    IF(
        AVERAGE('Dim Products'[List Price]) < AVERAGE('Dim Products'[Standard Cost]);
        AVERAGE('Dim Products'[List Price]);
        AVERAGE('Dim Products'[Standard Cost]
    ))
```

ISEMTPY(<Tabelle>) = “COUNTROWS(<Tabelle>) = 0”

SWITCH() = ein geschachteltes IF (mit einer Besonderheit, daß ein TRUE() als Expression angegeben werden kann und dann die Bedingungen in den einzelnen Parameter-Blöcken …)

```
SWITCH (Syntax Sugar 1) =
    SWITCH(
        SELECTEDVALUE('Parameter: Scale'[Scale]);
        "in EUR";1;
        "in Tsd";0,001;
        "in Mio";0,000001;
        "please select a scale"
    )
```

```
SWITCH (Syntax Sugar 2) =
    IF(SELECTEDVALUE('Parameter: Scale'[Scale]) = "in EUR";1;
        IF(SELECTEDVALUE('Parameter: Scale'[Scale]) = "in Tsd";0,001;
            IF(SELECTEDVALUE('Parameter: Scale'[Scale]) = "in Mio";0,000001;
            "please select a scale"
            )
        )
    )
```

TOTALYTD() = DATESYTD() = DATESBETWEEN() = FILTER()
SAMEPERIODLASTYEAR() = DATEADD() = DATESBETWEEN() = FILTER()

```
Sales YTD (Syntax Sugar 1) = TOTALYTD(
    [Sales];
    'Dim Date'[Date]
)
```

```
Sales LY (Syntax Sugar 1) = CALCULATE(
    [Sales];
    SAMEPERIODLASTYEAR('Dim Date'[Date])
)
```

```
Sales YTD (Syntax Sugar 2) = CALCULATE(
    [Sales];
    DATESYTD('Dim Date'[Date])
)
```

```
Sales LY (Syntax Sugar 2) = CALCULATE(
    [Sales];
    DATEADD('Dim Date'[Date];-1;YEAR)
)
```

```
Sales YTD (Syntax Sugar 3) = CALCULATE(
    [Sales];
    DATESBETWEEN(
        'Dim Date'[Date];
        DATE(YEAR(MAX('Dim Date'[Date]));1;1);
        MAX('Dim Date'[Date])
    )
)
```

DATESBETWEEN() ...
FILTER() ...

```
Sales YTD (Syntax Sugar 4) = CALCULATE(
    [Sales];
    FILTER(
        ALL('Dim Date'[Date]);
        'Dim Date'[Date] >= DATE(YEAR(MAX('Dim Date'[Date]));1;1)
        && 'Dim Date'[Date] <= MAX('Dim Date'[Date])
    )
)
```

CALCULATE: Filterbedingung ist ident mit FILTER()/ALL()-Kombination

```
Sales France (Syntax Sugar 1) = CALCULATE(
    [Sales];
    'Dim Customer'[Geo Country] = "France"
)
```

```
Sales France (Syntax Sugar 2) = CALCULATE(
    [Sales];
    FILTER(
        ALL('Dim Customer'[Geo Country]);
        'Dim Customer'[Geo Country] = "France"
    )
)
```

CALCULATE: KEEPFILTERS()/ALL()-Kombination ist ident mit VALUES()-Variante (genau genommen ist das aber kein Syntax Sugar sondern eine Syntax-Alternative)

```
Sales France mit Filter (Syntax Sugar 1) = CALCULATE(
    [Sales];
    KEEPFILTERS('Dim Customer'[Geo Country] = "France")
)
```

```
Sales France mit Filter (Syntax Sugar 2) = CALCULATE(
    [Sales];
    FILTER(
        VALUES('Dim Customer'[Geo Country]);
        'Dim Customer'[Geo Country] = "France"
    )
)
```

```
Sales France mit Filter (Syntax Sugar 3) = SUMX(
    FILTER(
        'Fact Sales';
        RELATED('Dim Customer'[Geo Country]) = "France"
    );
    'Fact Sales'[Sales Amount]
)
```

CALCULATE: REMOVEFILTERS() ist eine idente Funktion zu ALL() innerhalb von CALCULATE()

```
Sales in % absolute (Syntax Sugar) =
    DIVIDE(
        [Sales];
        CALCULATE(
            [Sales];
            REMOVEFILTERS('Fact Sales')
        )
    )
```

```
Sales in % absolute =
    DIVIDE(
        [Sales];
        CALCULATE(
            [Sales];
            ALL('Fact Sales')
        )
    )
```

SELECTEDVALUE() = HASONEVALUE() = DISTINCTCOUNT() = COUNTROWS(VALUES())

```
Selectedvalue (Syntax Sugar 1) =
    SELECTEDVALUE(
        'Dim Employees'[Employee Name];
        "please select an employee"
    )
```

```
Selectedvalue (Syntax Sugar 2) =
    IF(
        HASONEVALUE('Dim Employees'[Employee Name]);
        VALUES('Dim Employees'[Employee Name]);
        "please select an employee"
    )
```

```
Selectedvalue (Syntax Sugar 3) =
    IF(
        DISTINCTCOUNT('Dim Employees'[Employee Name])=1;
        VALUES('Dim Employees'[Employee Name]);
        "please select an employee"
    )
```

```
Selectedvalue (Syntax Sugar 4) =
    IF(
        COUNTROWS(VALUES('Dim Employees'[Employee Name]))=1;
        VALUES('Dim Employees'[Employee Name]);
        "please select an employee"
    )
```

SELECTEDVALUE() = HASONEVALUE() mit VALUES() oder FIRSTNONBLANK() in der SWITCH() Funktion

```
SWITCH (Syntax Sugar 1) =
    SWITCH(
        SELECTEDVALUE('Parameter: Scale'[Scale]);
        "in EUR";1;
        "in Tsd";0,001;
        "in Mio";0,000001;
        "please select a scale"
    )
```

```
SWITCH (Syntax Sugar 3) =
    IF(
        HASONEVALUE('Parameter: Scale'[Scale]);
        SWITCH(
            // FIRSTNONBLANK('Parameter: Scale'[Scale];'Parameter: Scale'[Scale]);
            // alter Syntax -> VALUES() reicht auch
            VALUES('Parameter: Scale'[Scale]);
            "in EUR";1;
            "in Tsd";0,001;
            "in Mio";0,000001
        );
        "please select a scale"
    )
```