e-Book

DAX mit Power BI

Webshop Ausgabe

Grundlagen und DAX Patterns für zahlreiche praktische Aufgabenstellungen

Ausgabe vom 05.05.2022

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 7
1.1 Ziele dieses e-Books 7
1.2 Nicht-Ziele 7
1.3 Sprachliche Konventionen 7
1.4 Power BI Desktop Version 8
1.5 Verwendete Shortcuts 8
2 Power BI Grundlagen 9
2.1 Was ist Power BI? 9
2.2 Power BI im Kontext des Microsoft Produktportfolios 10
2.3 Power BI Architektur und Fokus dieses e-Books 12
2.4 User Interface von Power BI Desktop 13
2.5 Wozu analytische Datenbanken? 14
2.6 Stufenmodell DAX & Datenmodellierung 15
2.7 Zentrale Begriffe im Datamodel 16
3 DAX Grundlagen 17
3.1 DAX Fundamentlas – Einstieg in die DAX Formelsprache 17
3.1.1 Aufgabenstellung 17
3.1.2 Lösung in Excel – SUMME(), SVERWEIS() und SUMMEWENN() 18
3.1.3 Lösung in Power BI 19
3.1.4 Charakteristiken der DAX Formelsprache 24
3.2 DAX Instrumente 25
3.2.1 Instrument "Measures" 26
3.2.2 Instrument "Calculated Columns" 28
3.2.3 Instrument "Calculated Tables" 29
3.2.4 Instrument "Row Level Security (RLS)" 30
3.2.5 Instrument "DAX Queries" 31
3.2.6 Abgrenzungen 32
3.3 DAX Evaluation Context 33
3.3.1 DAX Formula Evaluation 33
3.3.2 Konzept des Filter Context 34
3.3.3 Konzept des Row Context 34
3.3.4 Evaluation Matrix – Stufe 1 bis 3 35
3.3.5 Konzept der Context Transition 36
3.3.6 Konzept des Outer Filter Context und der Skalare 39
3.3.7 Konzept der Expanded Tables 40
3.3.8 Evaluation Matrix – Stufe 4 bis 6 42
3.4 DAX Syntax 43
3.4.1 Allgemeine DAX Syntax Regeln 43
3.4.2 Datentypen in DAX 43
3.4.3 Syntax Sugar 45
3.5 Zentrale DAX Funktionen 48
3.5.1 Aggregatorfunktionen 49
3.5.2 Datums- und Textfunktionen 50
3.5.3 Tabellenfunktionen 50
3.5.4 SELECTEDVALUE() 51

3.5.5 IF() vs. SWITCH() 51
3.5.6 BLANK() 52
3.5.7 CALCULATE() 52
3.5.8 Time Intelligence Funktionen 55
3.5.9 Iteratorfunktionen 57
3.5.10 FILTER() und weitere Tabellenfunktionen 59
3.5.11 Informationsfunktionen 61
3.5.12 Lookupfunktionen 62

4 DAX Patterns – Measures 65

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. = Distinctcount auf die Faktentabelle | DISTINCTCOUNT() 71
4.4.2 M04.2 Anzahl Länder, Produktgruppen, Monate, usw. = Distinctcount auf die Dimensionstabelle | DISTINCTCOUNT(), CALCULATE(), CROSSFILTER() 72

4.5 M05 Gefilterte Werte | CALCULATE() 74

4.5.1 M05.1 Gefilterte Werte – Allgemein | CALCULATE(), SUMX(), ALL(), VALUES(), KEEPFILTERS() und FILTER() 75
4.5.2 M05.2 Gefilterte Werte – IST-PLAN | CALCULATE() 79
4.5.3 M05.3 Gefilterte Werte – Wildcards | CALCULATE(), CONTAINSSTRING(), SEARCH(), CONTAINSSTRINGEXACT(), FIND() 80

4.6 M06 %-von-Gesamt Berechnungen | CALCULATE() 82

4.6.1 M06.1 %-von-Gesamt – Absolut | CALCULATE(), ALL(), REMOVEFILTERS() 83
4.6.2 M06.2 %-von-Gesamt – Relativ | CALCULATE(), ALL(Column) und ALLSELECTED(Column) 84
4.6.3 M06.3 %-von-Gesamt – Jahresweise | CALCULATE, ALLEXCEPT(Column) 86
4.6.4 M06.4 %- von-Umsatz Berechnung | CALCULATE(), FILTER(), ALLSELECTED() 87

4.7 M07 Time Intelligence Berechnungen | CALCULATE() 88

4.7.1 M07.1 Year-to-Date | TOTALYTD(), DATESYTD() 88
4.7.2 M07.2 Year-over-Year | SAMEPERIODLASTYEAR(), DATEADD() 91
4.7.3 M07.3 Period-over-Period | DATEADD(), PREVIOUSx(), PARALLELPERIOD() 92
4.7.4 M07.4 Week-over-Week mit Max-/Ebenen-Limitierung | SUMX(), FILTER(), LOOKUPVALUE(), HASONEVALUE() 93
4.7.5 M07.5 Change over Previous-N-Days mit Ebenen-Limitierung | DATESBETWEEEN(), HASONEVALUE() 96
4.7.6 M07.6 Moving-Annual-Total (MAT) mit Min-/Max-Limitierung | DATESINPERIOD() 97
4.7.7 M07.7 Moving-12-Months-Average | DATESINPERIOD() Expanded Tables 98
4.7.8 M07.8 Month-to-Date in Detaillisten | CALCULATE(), DATESMTD(), ALL(), ALLSELECTED(), DATESBETWEEN(), FILTER() 99
4.7.9 M07.9 Hervorheben der letzten 12 Monate | CALCULATE(), KEEPFILTERS(), DATESINPERIOD() 105

4.8 M08 Bestandslogik (Semi-Additive Measures) | CALCULATE() 107

4.8.1 M08.1 Bestandslogik für Bestandsgrößen 107
4.8.2 M08.2 Bestandslogik für Flußgrößen 111

4.9 M09 Iteratoren 113

4.9.1 M09.1 Menge x Preis zur Umsatzermittlung | SUMX(), RELATED() 113
4.9.2 M09.2 Verdurchschnittung vervielfachter Werte: Anzahl Paxe | SUMX(), VALUES(), AVERAGE(), CALCULATE() … ADDCOLUMNS(), SUMMARIZE() 114
4.9.3 M09.3 Verdurchschnittung vervielfachter Werte: Durchschnittsalter | AVERAGEX(), VALUES(), AVERAGE(), CALCULATE() 117
4.9.4 M09.4 Größte/kleinste/durchschnittliche Tagesumsätze pro Monat | MAXX(), MINX(), AVERAGEX(), CALCULATE() 118
4.9.5 M09.5 Korrekte Ermittlung auf Total-Ebene | SUMX() 120

4.9.6 M09.6 Anzeige der selektierten Werte als Textkette | CONCATENATEX(), VALUES(), CROSSFILTER(), SUMMARIZE() 121

4.10 M10 Rankings und TOP-N Auswertungen | RANKX() und TOPN() 123

4.10.1 M10.1 Basislösung RANKX() | RANKX(), ALL(), ALLSELECTED(), HASONEVALUE() 123

4.10.2 M10.2 Basislösung TOPN() | TOPN(), ALL() 127

4.11 M11 Parameter Table Lösungen 128

4.11.1 M11.1 Measure Einfach-Selektion | DATATABLE(), SELECTEDVALUE(), SWITCH() 128

4.11.2 M11.2 Measure Mehrfach-Selektion | DATATABLE(), CONTAINS() 129

4.11.3 M11.3 Tausender-Umrechnung | DATATABLE(), SELECTEDVALUE() 130

4.11.4 M11.4 Dynamische Moving-N-Month-Averages | GENERATESERIES(), SELECTEDVALUE() 131

4.11.5 M11.5 Parametrisiertes Ranking | GENERATESERIES(), SELECTEDVALUE(), IF(), SWITCH() 133

4.11.6 M11.6 Parametrisiertes Zeitfenster | GENERATESERIES(), SELECTEDVALUE(), CALCULATE(), KEEPFILTERS(), DATESINPERIOD() 137

4.11.7 M11.7 Parametrisiertes Wirtschaftsjahr | DATATABLE(), SELECTEDVALUE(), SWITCH(), TOTALYTD() 140

4.12 M12 Farb-, Text- und Bild-Measures 143

4.12.1 M12.1 Farb-Measures für dynamische Formatierung | SWITCH() 143

4.12.2 M12.2 Text-Measures für dynamische Titel | CONCATENATEX(), VALUES(), MIN(), MAX() 144

4.12.3 M12.3 Text-Measures für die bedingte Anzeige einer Selektion | CALCULATE(), MAX(), ALL() 145

4.12.4 M12.4 Text-Measures für dynamische Indikatoren, Ratings und Sortierung | SWITCH(), UNICHAR(), FORMAT(), REPT() 146

4.12.5 M12.5 Bild-Measures für dynamische Indikatoren | SWITCH() 149

4.13 Weitere Patterns 150

5 DAX Patterns – Calculated Columns 151

5.1 C01 Preisberechnung auf Datensatzebene (Erlös/Menge) | DIVIDE() 152

5.2 C02 Attribute in Datumsdimension hinzufügen | Datumsfunktionen YEAR(), QUARTER(), MONTH(), WEEKNUM(), WEEKDAY() und FORMAT() 152

5.3 C03 Textfelder zerlegen oder verketten | Textfunktionen LEFT(), RIGHT(), MID(), FIND() und CONCATENATE() 153

5.4 C04 Statische Altersberechnung 154

5.4.1 C04.1 Altersberechnung in der Dimensionstabelle 154

5.4.2 C04.2 Altersberechnung in der Faktentabelle | IF(), AVERAGE() 159

5.5 C05 Auflösen von Parent-Child Strukturen | PATH(), PATHITEM(), LOOKUPVALUE() 160

5.6 C06 Lookups in verknüpften Tabellen 162

5.6.1 C06.1 Lookup in verknüpfter Dimensionstabelle ("SVERWEIS mit aktiver Beziehung") | RELATED() 162

5.6.2 C06.2 Lookup in verknüpfter Faktentabelle | RELATEDTABLE() und COUNTROWS() 163

5.6.3 C06.3 Bedingter Lookup in verknüpfter Faktentabelle | FILTER(), CALCULATETABLE(), COUNTROWS() 164

5.7 Weitere Patterns 165

6 DAX Patterns – Calculated Tables 166

6.1 T01 Hard-codierte Tabelle | Table Constructor und DATATABLE() 167

6.2 T02 Berechnete Parameter Tabelle | GENERATESERIES() 169

6.3 T03 Berechnete Datumsdimension 171

6.3.1 T03.1 Erstellen der Datumsspalte | CALENDAR() und CALENDARAUTO() 171

6.3.2 T03.2 Attribute für die Datumsdimension | ADDCOLUMNS() 173

6.4 T04 Duplizieren einer bestehenden Tabelle | ALL() und ALLNOBLANKROW() 174

6.5 T05 Extraktion der Unique Werte aus einer Spalte 175

6.5.1 T05.1 Ungefilterte Liste an unique Werten | ALL(), ALLNOBLANKROW(), VALUES() und DISTINCT() 175

6.5.2 T05.2 Gefilterte Liste an unique Werten | FILTER() und CALCULATETABLE() 176

6.6 T06 Extraktion der Unique Werte aus mehreren Spalten | ALL(), ALLEXCEPT() und SUMMARIZE() 177

6.7 Weitere Patterns 179

7 DAX Patterns – Row Level Security 180

7.0 Allgemeines zur Row Level Security 181

7.1 R01 Statische RLS 185

7.1.1 R01.1 Statische RLS mit UND-Bedingungen | (Filterbedingungen) 185

7.1.2 R01.2 Statische RLS mit ODER-Bedingungen | (Filterbedingungen) 187

7.2 R02 Dynamische RLS mit 1:n Beziehung 188

7.2.1 R02.1 Dynamische RLS mit 1:n und Top Management Role | USERPRINCIPALNAME() 188

7.2.2 R02.2 Dynamische RLS mit 1:n und Top Management Flag | USERPRINCIPALNAME(), MAXX(), FILTER() 189

7.3 R03 Dynamische RLS mit m:n Pattern 190

7.3.1 R03.1 Dynamische RLS mit m:n bidirektionaler Beziehung | USERPRINCIPALNAME() 190

7.3.2 R03.2 Dynamische RLS mit m:n Filterbedingung | USERPRINCIPALNAME(), IN(), SELECTCOLUMNS(), FILTER() 192

7.3.3 R03.3 Dynamische RLS mit m:n Lookup Column | USERPRINCIPALNAME(), NOT(), ISBLANK(), LOOKUPVALUE() 194

7.4 Weitere Patterns 196

8 Informationen zu den Trainingscases 197

8.1 Trainingscase „AW Reseller Sales" 197

8.2 Trainingscase „Headcount Reporting" 198

8.3 Trainingscase „HR Reporting" 199

8.4 Trainingscase „Multi-Fakten Schema Typ 1" 199

8.5 Trainingscase „P&L Reporting" 200

8.6 Trainingscase „Projekte aus Zeiträumen" 201

8.7 Trainingscase „Sales Forecast" 201

8.8 Trainingscase „Sales Report" 202

8.9 Trainingscase „Semi-additive Kennzahlen" 202

8.10 Trainingscase „Simple Tabelle" 202

8.11 Trainingscase „Survey Data" 203

8.12 Trainingscase „Vervielfachte Werte" 203

9 Cheat Sheet & Best Practices 204

10 Quellen 205

10.1 DAX Referenzen 205

10.2 Bücher und Blogs 206

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 2022**
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] + [Space]	**Intellisense** im DAX-Editor aufrufen
[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?

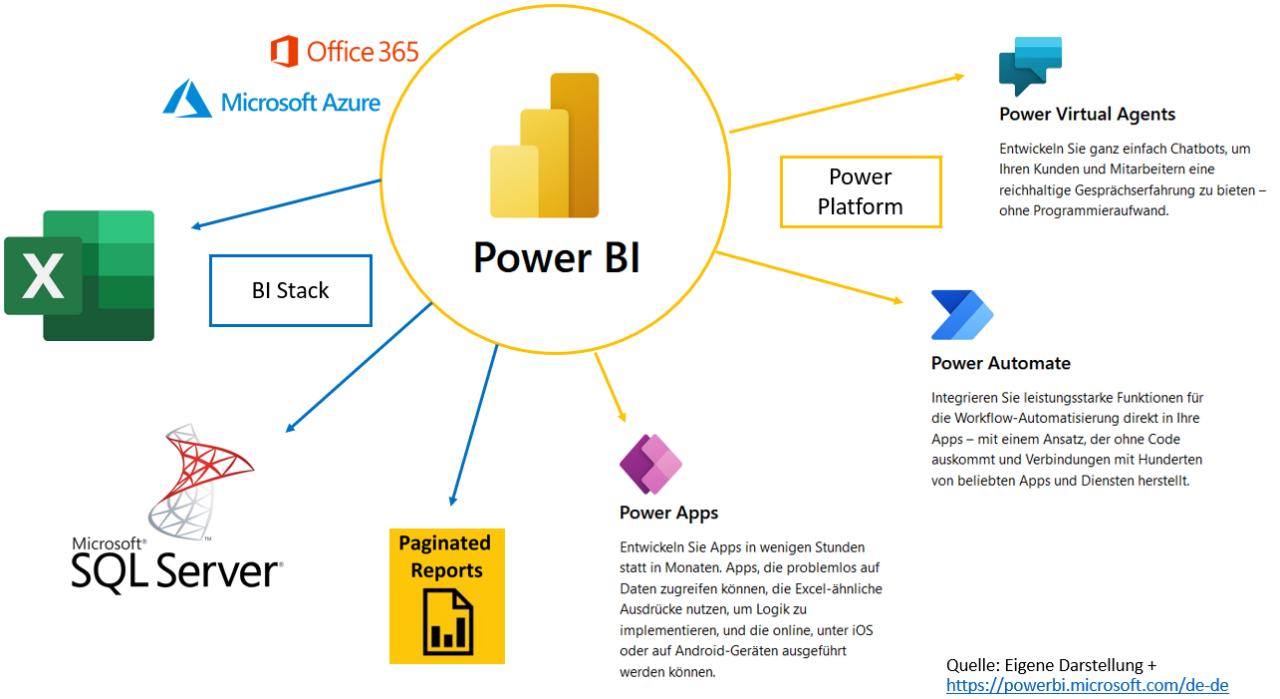
Power BI ist ein cloud-basiertes Dashboarding Tool mit einer sehr leistungsfähigen Query- und Datenmodell-Komponente als „Engine". In Power BI werden sogenannte analytische Datenmodelle aus verschiedenen Datenquellen aufgebaut, die Auswertung erfolgt – vergleichbar mit Power Point – auf einem Canvas.

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

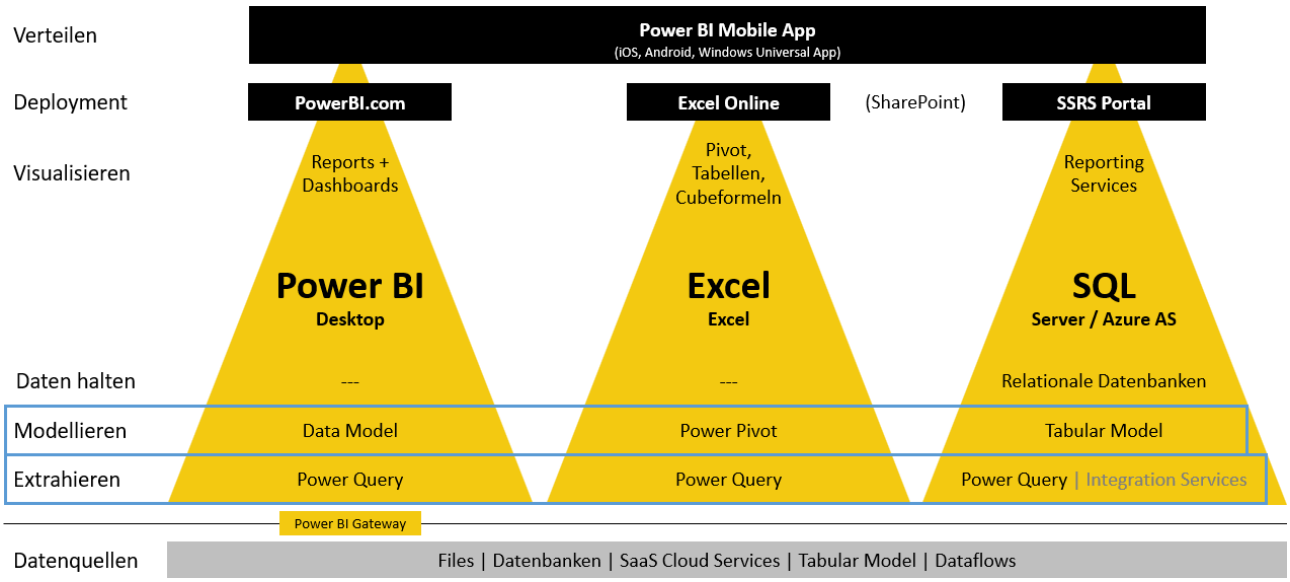
1. **Power BI** ist das Tool für Dashboarding und mobiles Reporting
2. **Excel** ist das Tool für die adhoc Analyse und Simulation.

   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.

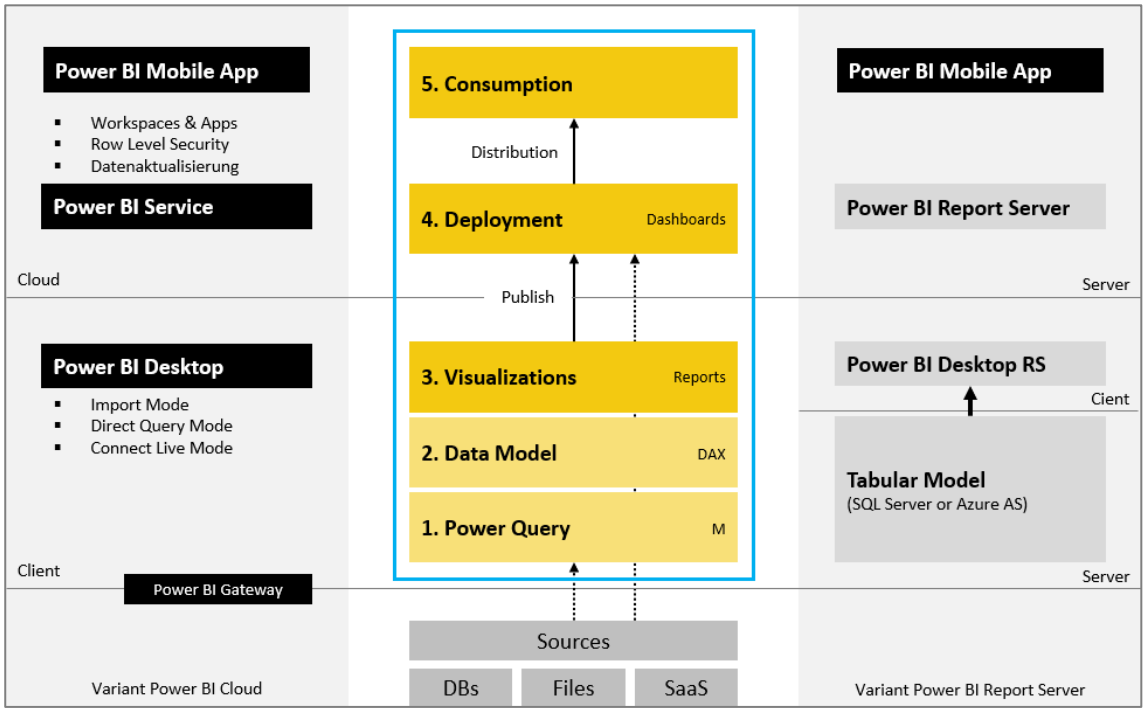


Diese Technologien haben in den 3 Produkten einige (sinnvolle) Spezifika, unterscheiden sich unglücklicherweise aber auch im Entwicklungsstand. Excel Power Pivot wird seit einigen Jahren nicht mehr weiterentwickelt, während das Power BI Datamodel und das SQL Server Tabular Model sich auf einem hohen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

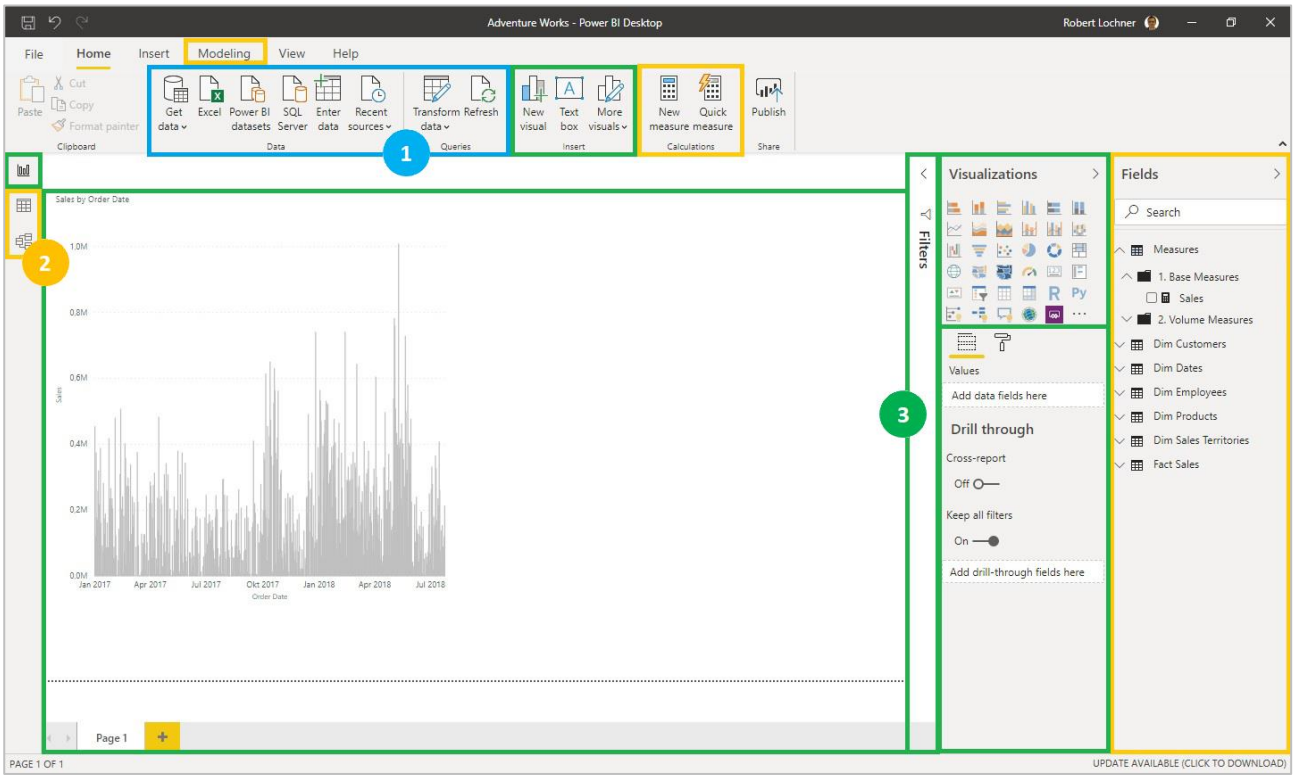


Der Fokus dieses e-Books liegt auf der Komponente „2. Data Model".

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 WOZU ANALYTISCHE DATENBANKEN?

Analytische Datenbanken sind optimiert für die performante und intuitive **Auswertung von Datentabellen durch Fachanwender** (und nicht durch die IT/Developer):

- Im Gegensatz zu Excel sind sämtliche Daten in Tabellen strukturiert, die Datenmengen können viel größer sein und die Auswertung ist sehr intuitiv auch ohne Formel-Know-How („drag & drop") möglich.
- Im Gegensatz zu relationalen Datenbanken („SQL") sind diese intuitiv per Drag & Drop („Slice & Dice") und sehr performant auswertbar. Vergleichbar ist hingegen die tabellarische Struktur und die Beziehungen zwischen den Tabellen, letztere unterscheiden sich in ihrem Charakter aber grundlegend von den Beziehungen in einer relationalen Datenbank.
- Die Anbindung und Kombination von verschiedenen Datenquellen ist sehr einfach. Im Gegensatz zu Excel sind Copy & Paste Lösungen nicht möglich, sämtliche Daten sind aus Datenquellen angebunden („Knopfdrucklösung" zur laufenden Aktualisierung).

Die Tabellen in einer analytischen Datenbank wie Power BI werden zu analytischen Datenmodellen kombiniert. Ein analytisches Datenmodell ist eine Einheit von Datentabellen zu einem bestimmten Thema (bspw. Sales Reporting, HR Reporting, usw.). Im einfachsten Fall besteht das analytische Datenmodell nur aus einer einzigen Tabelle, anspruchsvollere Datenmodelle bestehen aus mehreren oder vielen Tabellen, die idR durch Filterbeziehungen miteinander verbunden sind (fallweise aber auch bewußt nicht verbunden werden).

Wozu mit analytischer Datenmodellierung in Power BI beschäftigen?

- **Motivation #1: Informationen aus den Daten holen**

  Ein gutes Datenmodell lässt umfangreichere Auswertungen zu als ein schlecht/falsch konzipiertes Datenmodell. Je komplexer das Datenmodell (bspw. Multi-Fakten Schema), umso mehr DAX Know-How ist für bereits relativ einfache Anforderungen notwendig. Je simpler das Datenmodell (Flat Table, Star Schema), umso weniger DAX Know-How ist notwendig bzw. umso mehr kann mit dem bestehenden Know-How aus den Daten geholt werden.

- **Motivation #2: Interaktivität in den Reports erhöhen**

  Das Datenmodell ist für die Interaktivität der Reports ausschlaggebend (Filterzusammenhänge über Beziehungen).

- **Motivation #3: Performance der Reports**

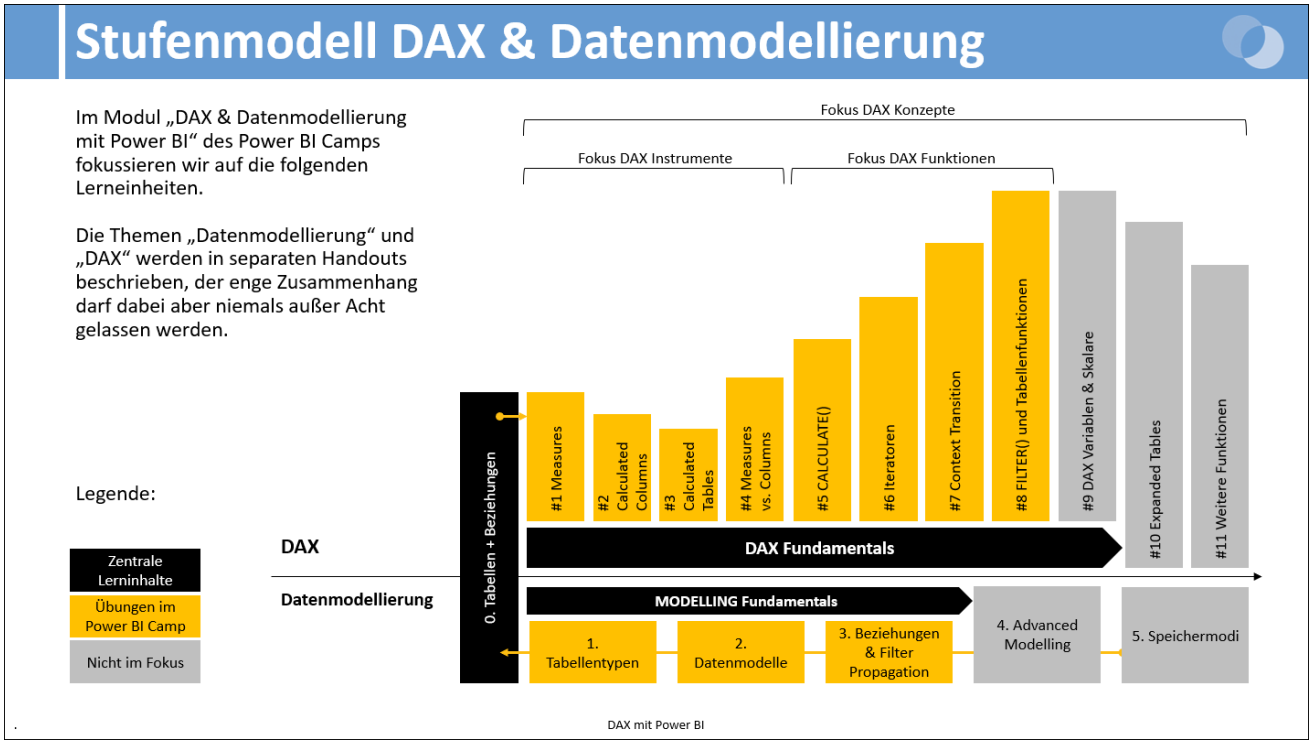
  Das Datenmodell bestimmt die Performance der Reports. Bei kleinen Datenmengen mit einfachen Berechnungen (bspw. Summen) spielt das Datenmodell keine nennenswerte Rolle, bei großen Datenmengen mit anspruchsvollen Berechnungen (bspw. Iteratoren) ist das Datenmodell entscheidend für die Performance.

"DAX is the key, the Datamodel is the basis."

„Insane DAX, necessary for a messy datamodel."

2.6 STUFENMODELL DAX & DATENMODELLIERUNG

Zum Lernen der DAX Formelsprache und der Modellierungsthemen in Power BI hat sich das folgende Stufenmodell bewährt: versteht man ein Thema nicht (mehr), dann bleibt man einfach auf der Stufe davor stehen … solange bis das Wissen ausreichend gefestigt ist und die weitere Vertiefung dann individuell möglich wird.



Die Themen „Datenmodellierung“ und „DAX“ werden in separaten Handouts beschrieben, der enge Zusammenhang darf dabei aber niemals außer Acht gelassen werden.

2.7 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"
= die Visuals in Power BI erhalten über DAX Queries die Daten aus dem Datenmodell

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

3 DAX GRUNDLAGEN

3.1 DAX FUNDAMENTLAS – 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

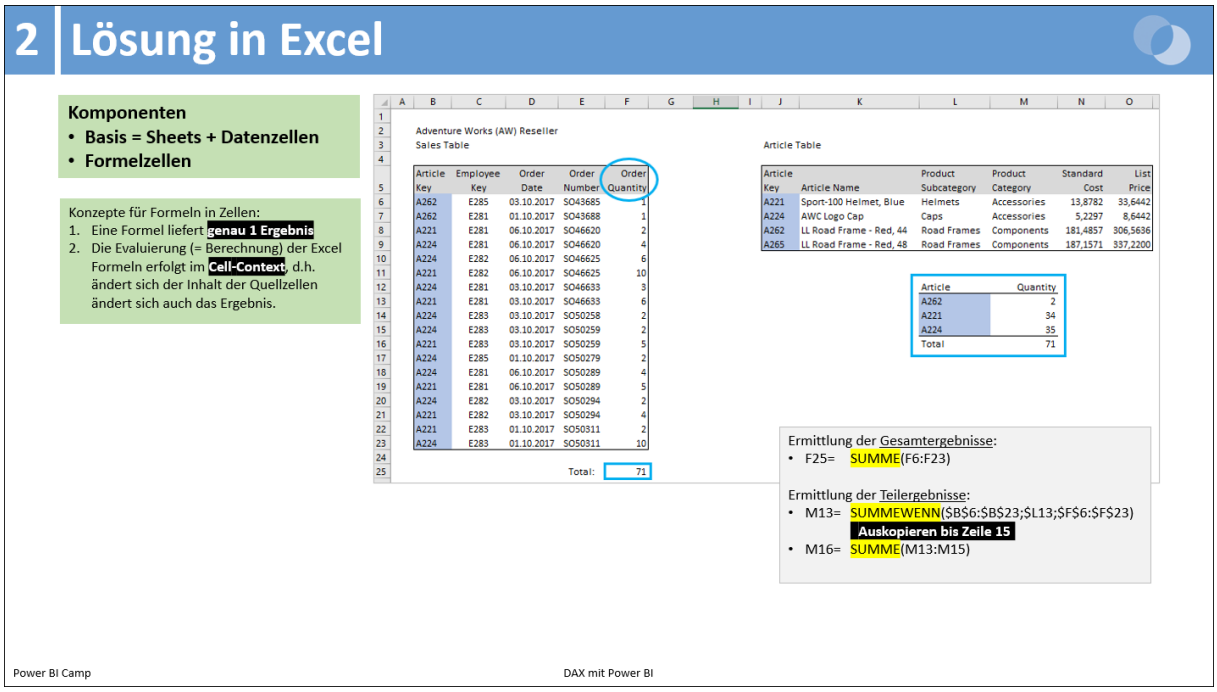
Article Table

Article Key	Article Name	Product Subcategory	Product Category	Standard Cost	List Price
A221	Sport-100 Helmet, Blue	Helmets	Accessories	13,8782	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

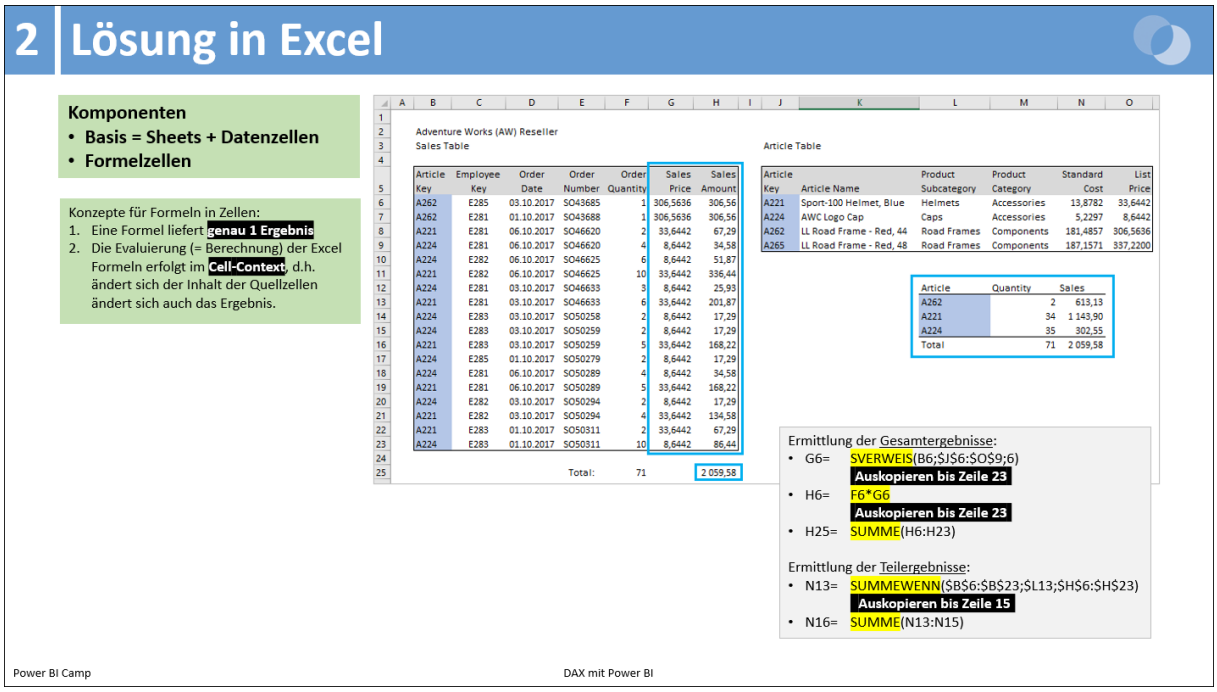
Power BI Camp DAX mit Power BI

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". In den Formelzellen kommen die Funktionen SUMME() und SVERWEIS() …

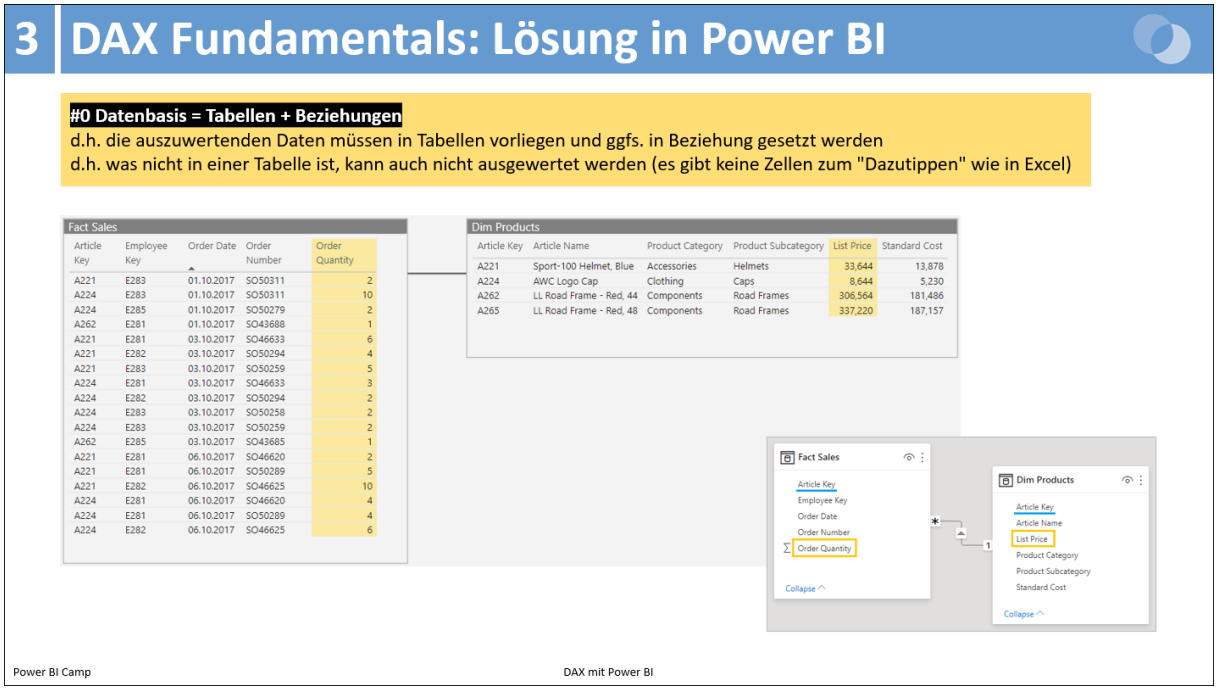


… sowie SUMMEWENN() zum Einsatz:



3.1.3 LÖSUNG IN POWER BI

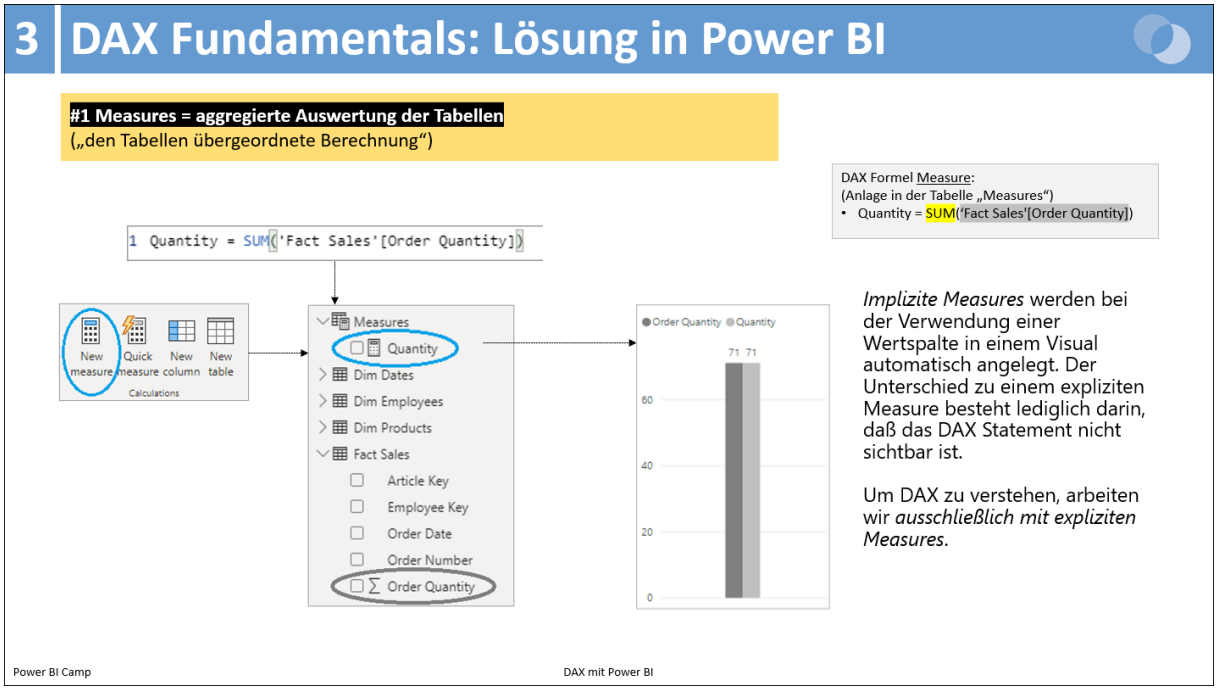
In Power BI werden **sämtliche Daten in Tabellen gespeichert**, die über **Filterbeziehungen** verbunden sein können (aber nicht müssen). Das vorliegende Beispiel kann als „Fact Sales" und „Dim Products" in Power BI tabellarisch abgebildet werden, die beiden Tabellen sind über eine Filterbeziehung zu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

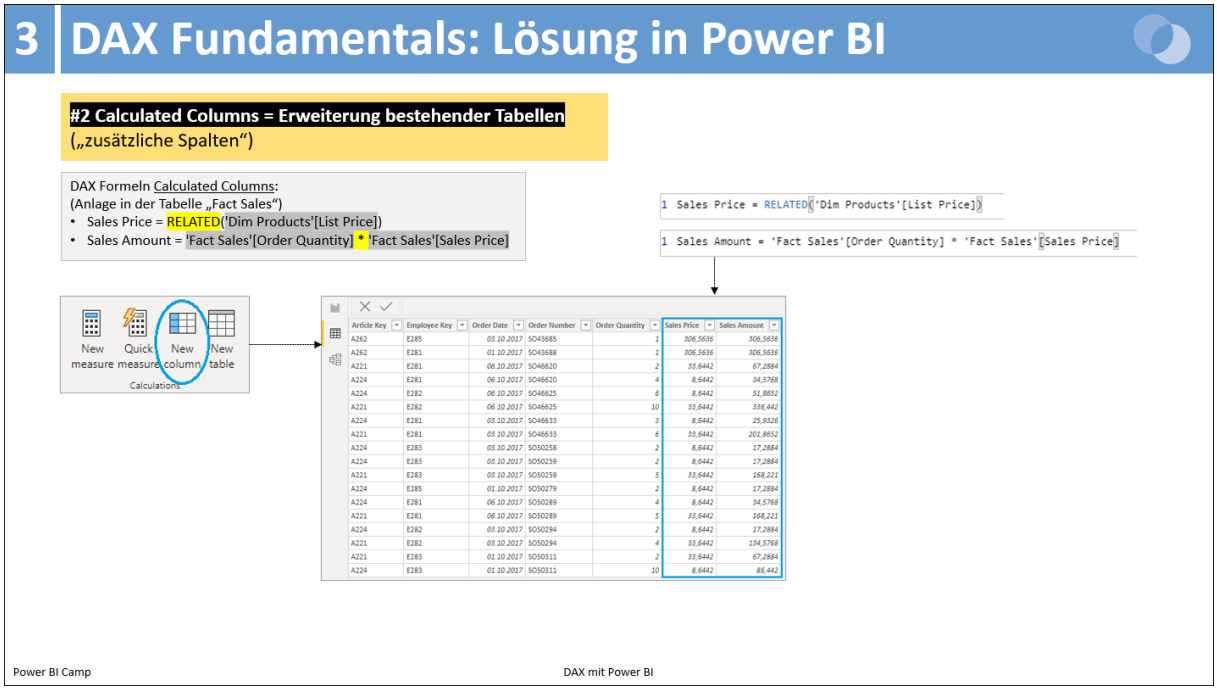


Zur Ermittlung des Gesamtumsatzes werden in Power BI (ähnlich wie in Excel) zwei **Calculated Columns** angelegt, um den **List Price** aus der Dimensionstabelle in die Faktentabelle zu holen und dort mit der **Quantity** zur **Sales Amount** zu multiplizieren:

```
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:



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

Konzepte für Calculated Columns:

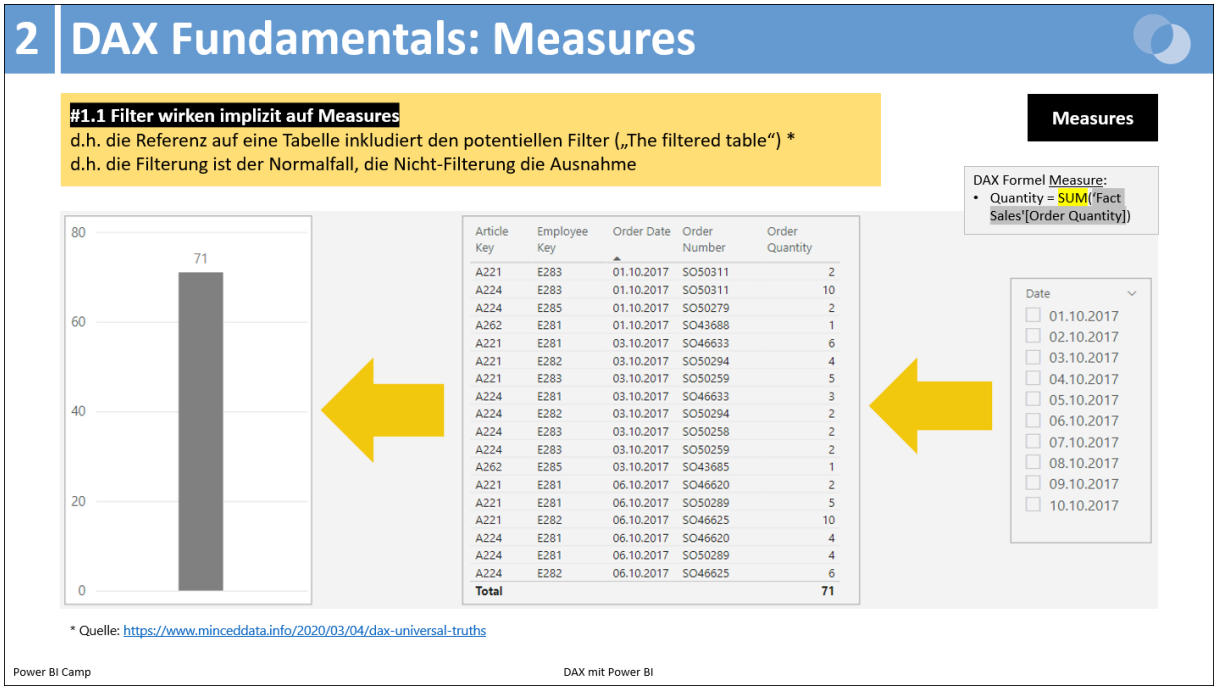
a. Eine einzige Formel liefert so **viele Ergebnisse**, wie es Datensätze in der Tabelle gibt.

b. Die Evaluierung (= Berechnung) einer Calculated Column erfolgt im **Row Context**, d.h. die Ermittlung erfolgt im Kontext der jeweiligen Zeile der Tabelle.

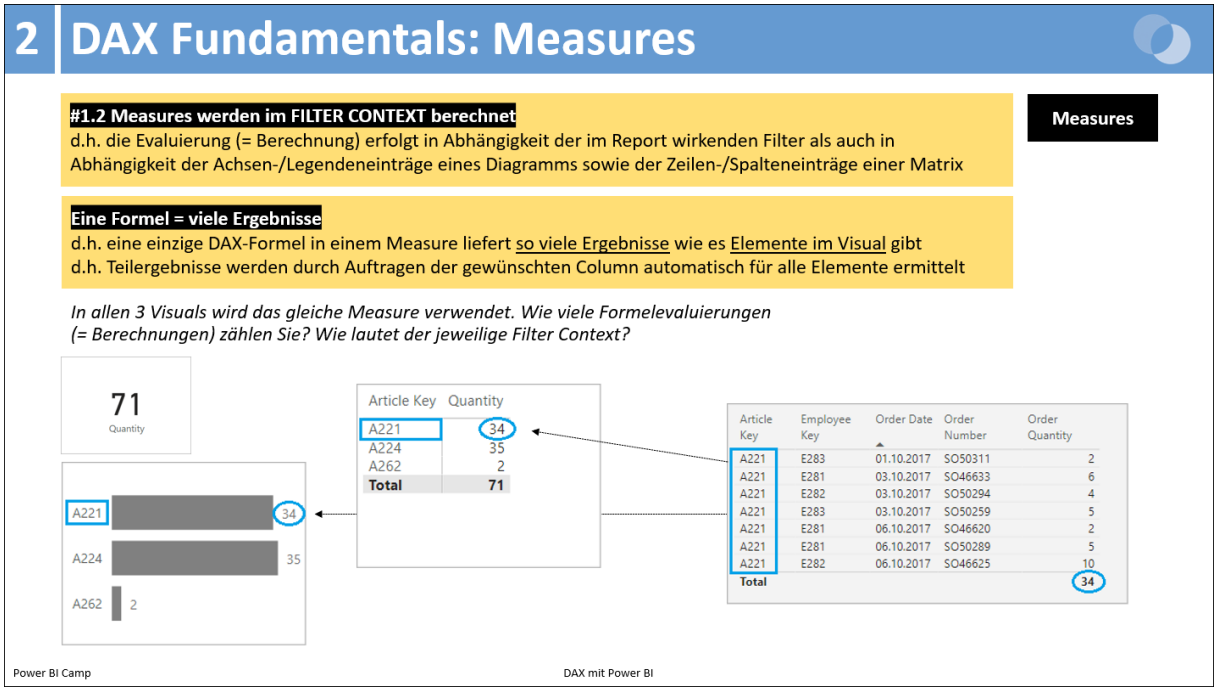
   Frage: wie oft wurden die beiden DAX-Formeln der Calculated Column im Screenshot oben evaluiert?

Zurück zu den Measures.

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



Measures werden im sog. Filter Context berechnet:



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:

2 DAX Fundamentals: Measures

#1.3 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

2 DAX Fundamentals: Measures

#1.4 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**	

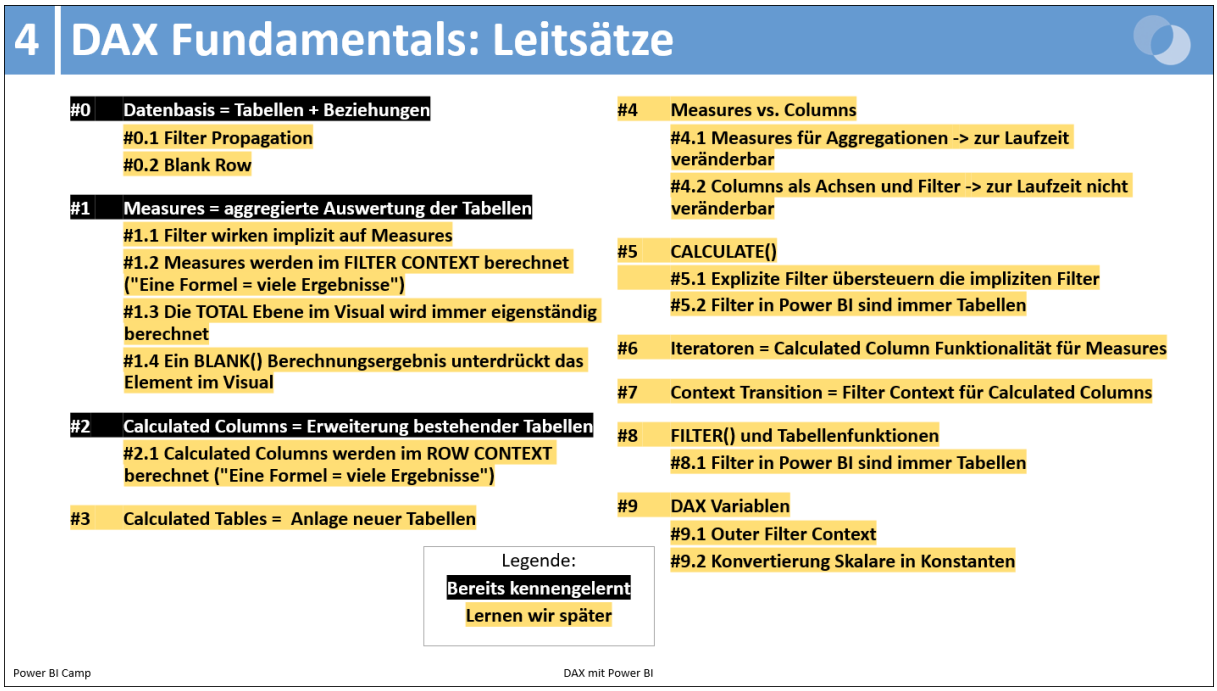
Article Key	Demo BLANK
A221	yes
A224	yes
Total	

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

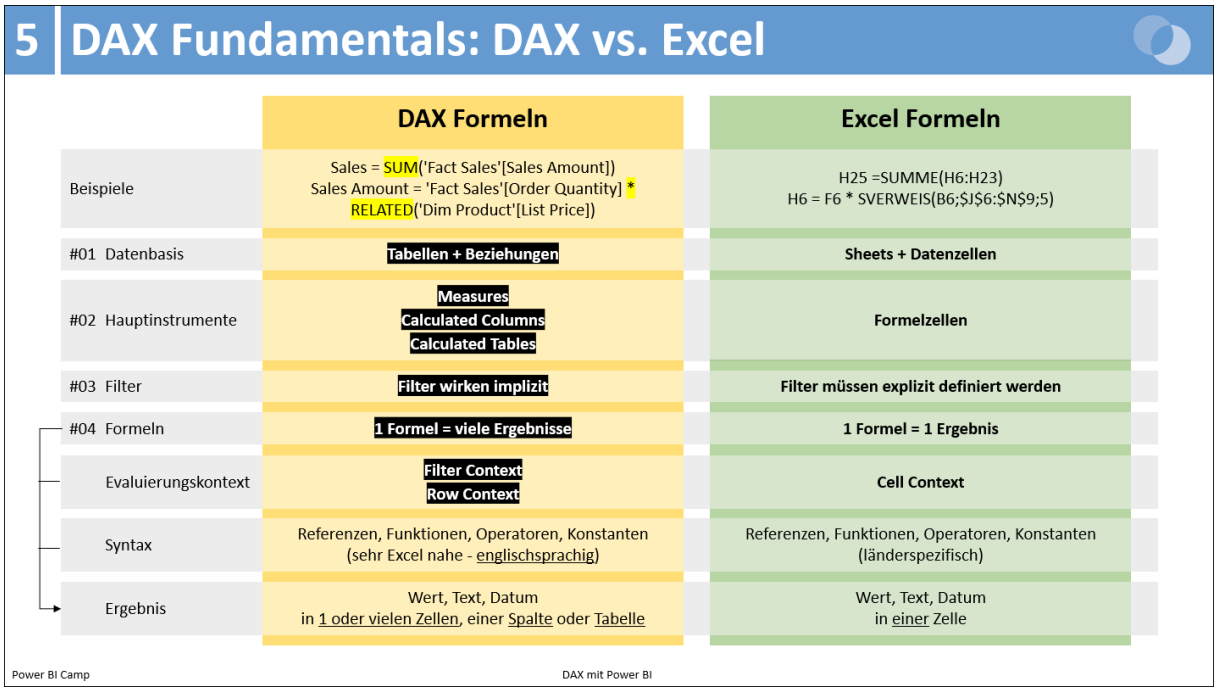
Als erste Zusammengefasst ergeben sich folgende **DAX Fundamentals**, diese Leitsätze helfen uns dabei, komplexe Fragestellungen zu lösen bzw. DAX Lösungsbausteine rasch erklären zu können:



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.



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

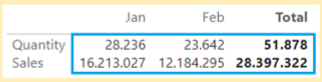
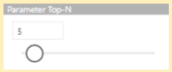
Weiterführend:
https://www.quora.com/Why-do-all-the-BI-consultants-hate-DAX/

3.2 DAX INSTRUMENTE

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:

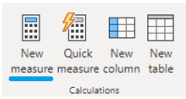
	MEASURES	Calculated COLUMNS	Calculated TABLES
Beispiel	Sales = SUM('Fact Sales'[Sales Amount])	Sales Price = RELATED('Dim Product'[List Price])	Parameter Tabelle = GENERATESERIES(1;10;1)
Ausführungsebene	"Überhalb" aller Tabellen (Syntax: Aggregationsfunktion zwingend nötig, lediglich bei Iteratoren 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) Aggregation benötigt wird** 	**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**		

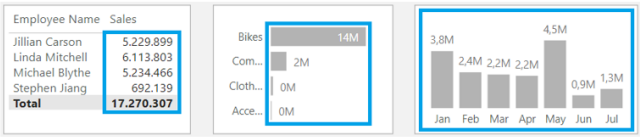
Hinweis: in Excel können Formeln auch in der Bedingten Formatierung, in den Spalten eines Tabellenobjekts sowie in PivotTables/PivotCharts eingegeben werden. Für den Vergleich hier wird aber lediglich das Kerninstrument „Zellen" betrachtet.

3.2.1 INSTRUMENT "MEASURES"

Was sind 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 außerhalb der Tabellen erfolgt, kann keine direkte Referenz auf Columns gesetzt werden, sondern müssen Columns **immer mit Aggregationsfunktionen** umschlossen werden (Ausnahme: Iterator-Funktionen)

Automatismen/Wizards:
- Implizite Measures
- Quick Measures

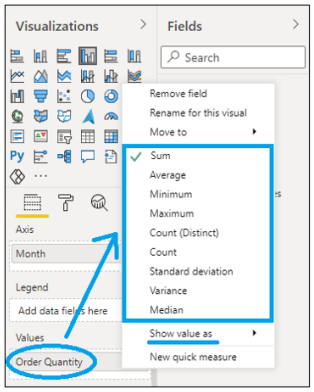
Power BI Camp DAX mit Power BI

Zusätzliche Eigenschaften:
- Benötigen keinen RAM-Speicher jedoch CPU-Rechenleistung.

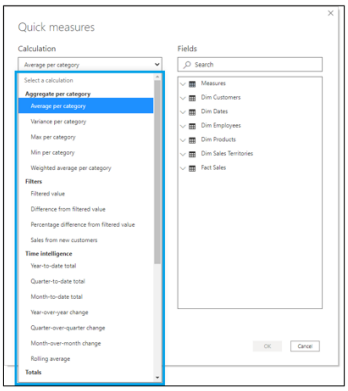
Automatismen/Wizards:

Wizards für 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:

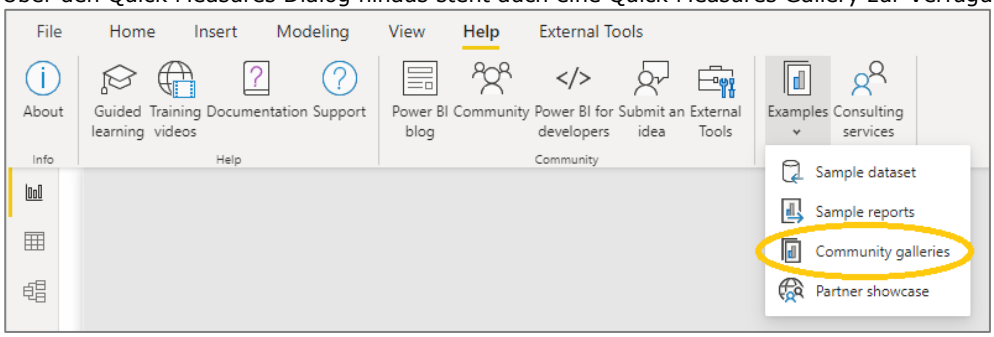


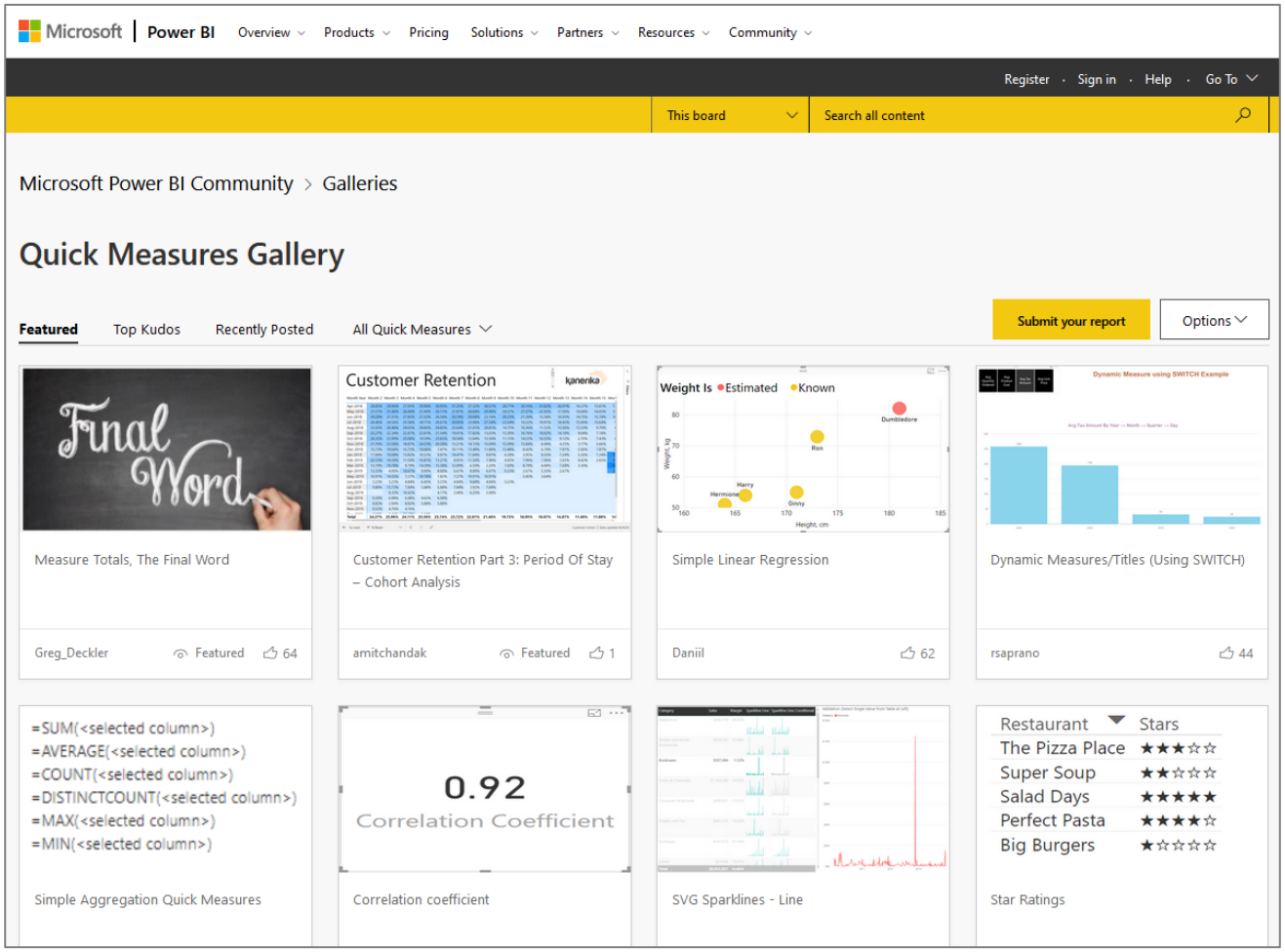
- **Quick Measures**
  Für anspruchsvollere Berechnungen stehen die sogenannten Quick Measures zur Verfügung. Für Einsteiger in DAX sind die erzeugten DAX Statement aber häufig unnötig kompliziert und daher nicht immer gut als Lernvorlage geeignet.



Power BI Camp DAX mit Power BI

Über den Quick Measures Dialog hinaus steht auch eine Quick Measures Gallery zur Verfügung:

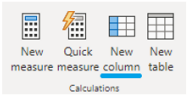




3.2.2 INSTRUMENT "CALCULATED COLUMNS"

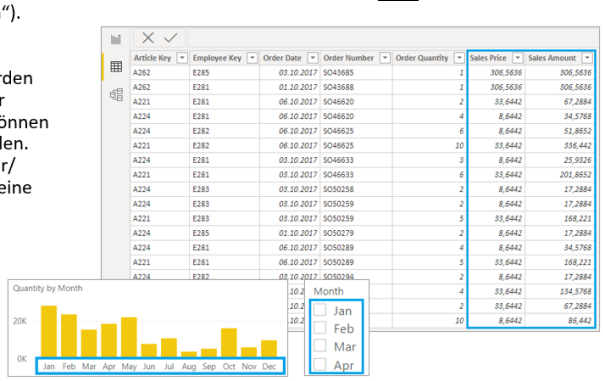
Was sind Calculated Columns?

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.



Automatismen/Wizards:
- Grouping & Binning
- Clustering

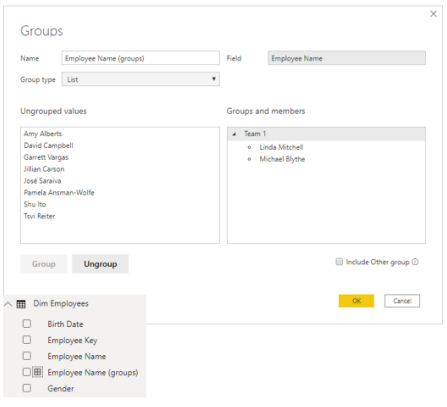
Power BI Camp DAX mit Power BI

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

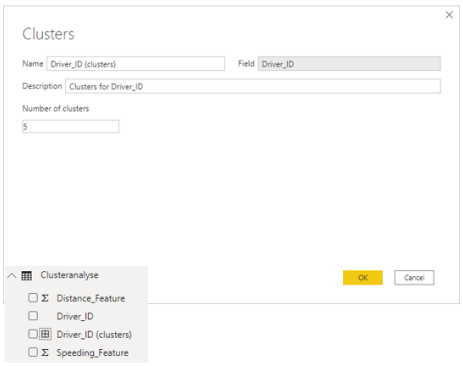
Automatismen/Wizards:

Wizards für 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"

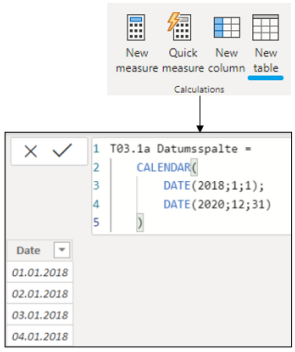
Was sind Calculated Tables?

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

Automatismen/Wizards:
- What-If Parameter Tabellen
- Auto Date/Time Dimension



Power BI Camp DAX mit Power BI

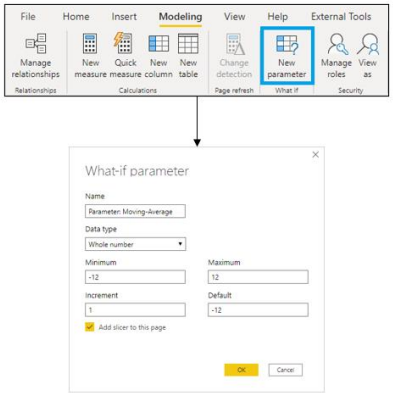
Zusätzliche Eigenschaften:

- 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.

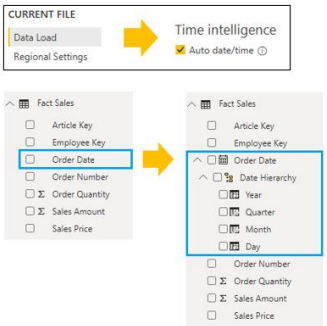
Automatismen/Wizards:

Wizards für Calculated Tables

Automatismen/Wizards:
- What-If Parameter Tabellen
  Aufruf erfolgt über Button im Menü



- **Auto Date Dimension für Datumsfelder**
  Aktivierung über Optionen & Settings



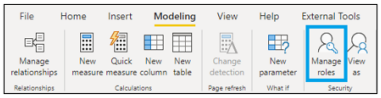
Power BI Camp DAX mit Power BI

3.2.4 INSTRUMENT "ROW LEVEL SECURITY (RLS)"

Instrument „Row Level Security“

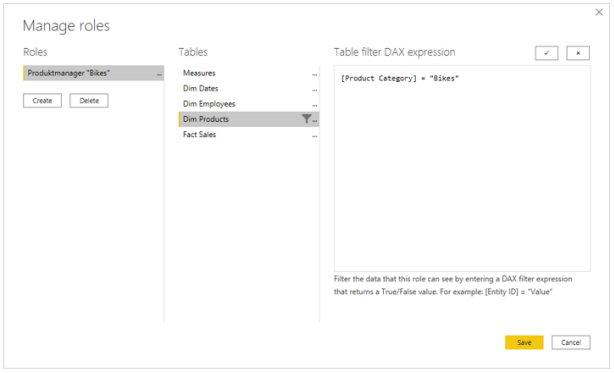
Einsatzgebiete:

- **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

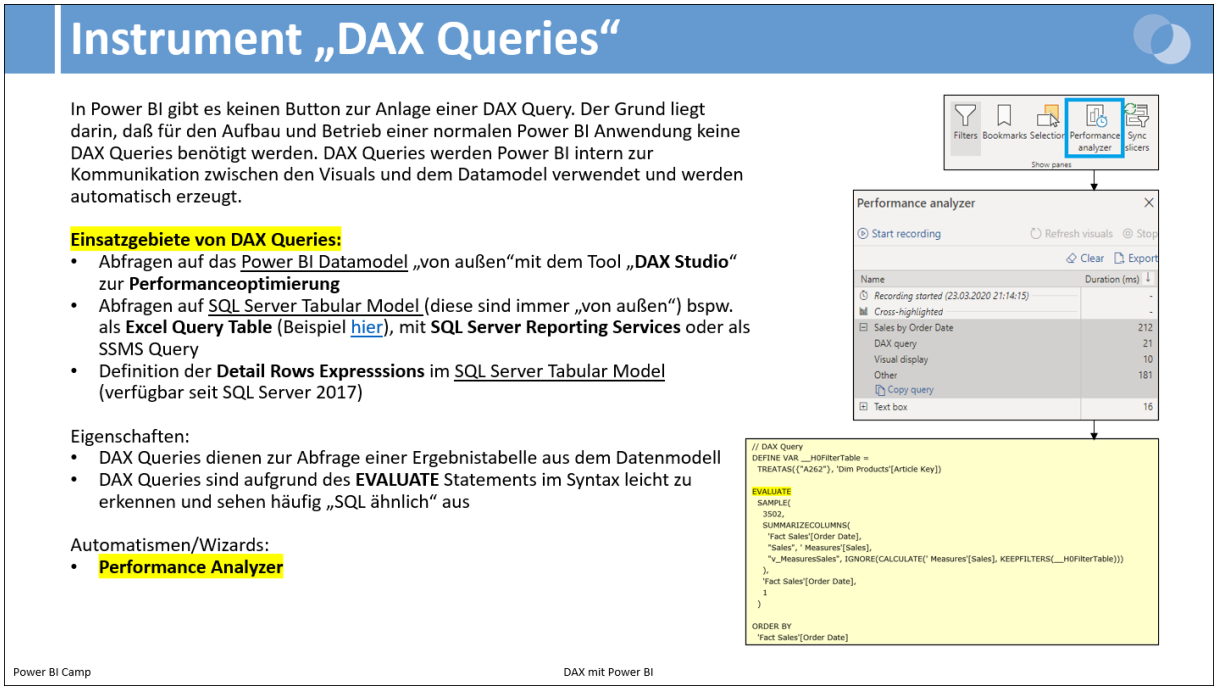
Weiterführend:

- 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"

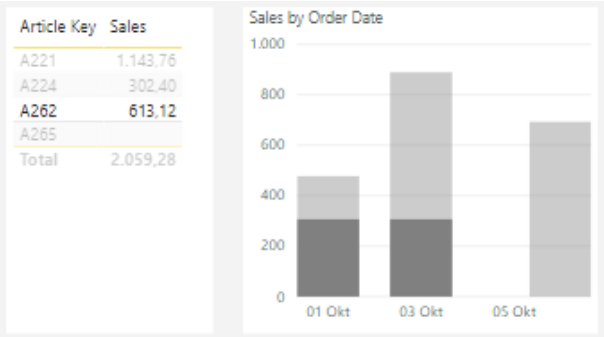


Automatismen/Wizards:

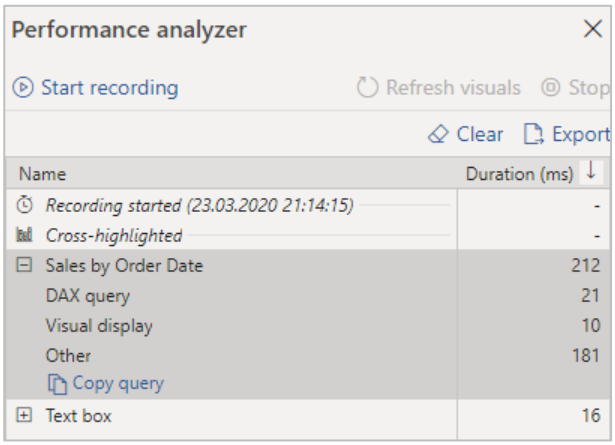
- Performance Analyzer

  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 oder mit der DAX-Funktion UNION() 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

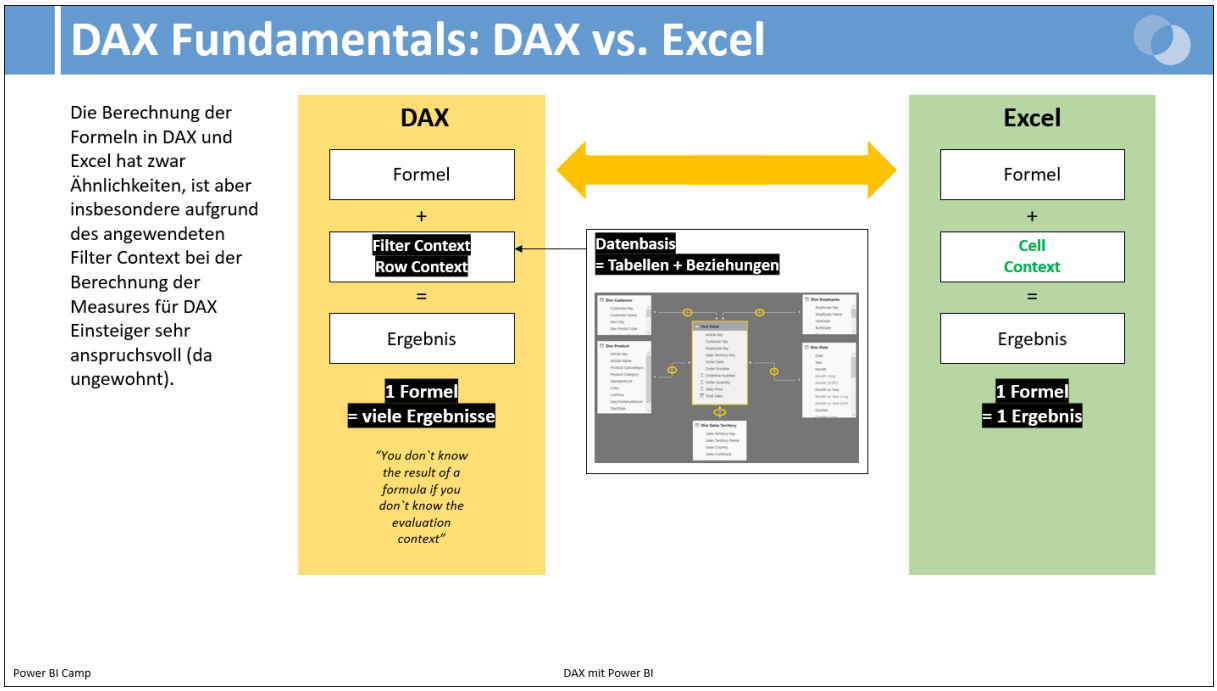
3.3 DAX EVALUATION CONTEXT

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.

Das Ergebnis einer Berechnung hängt 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

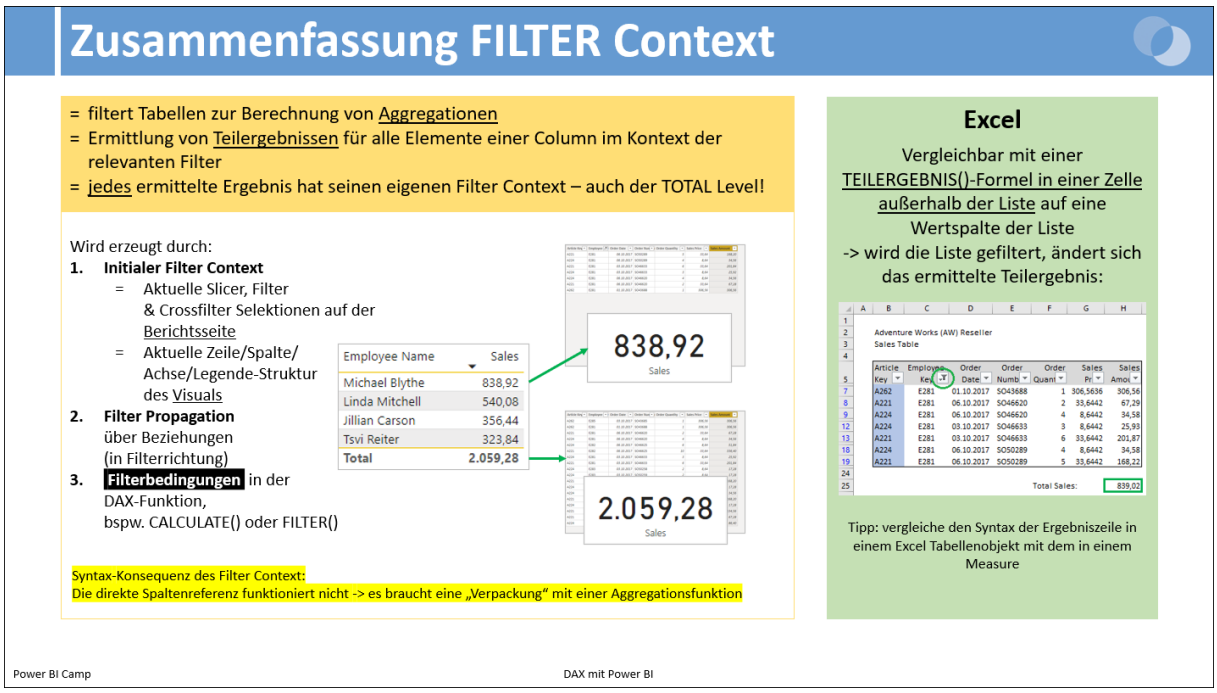


Zur Berechnung eines konkreten Ergebnisses werden folgende Evaluierungsschritte angewendet:

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/

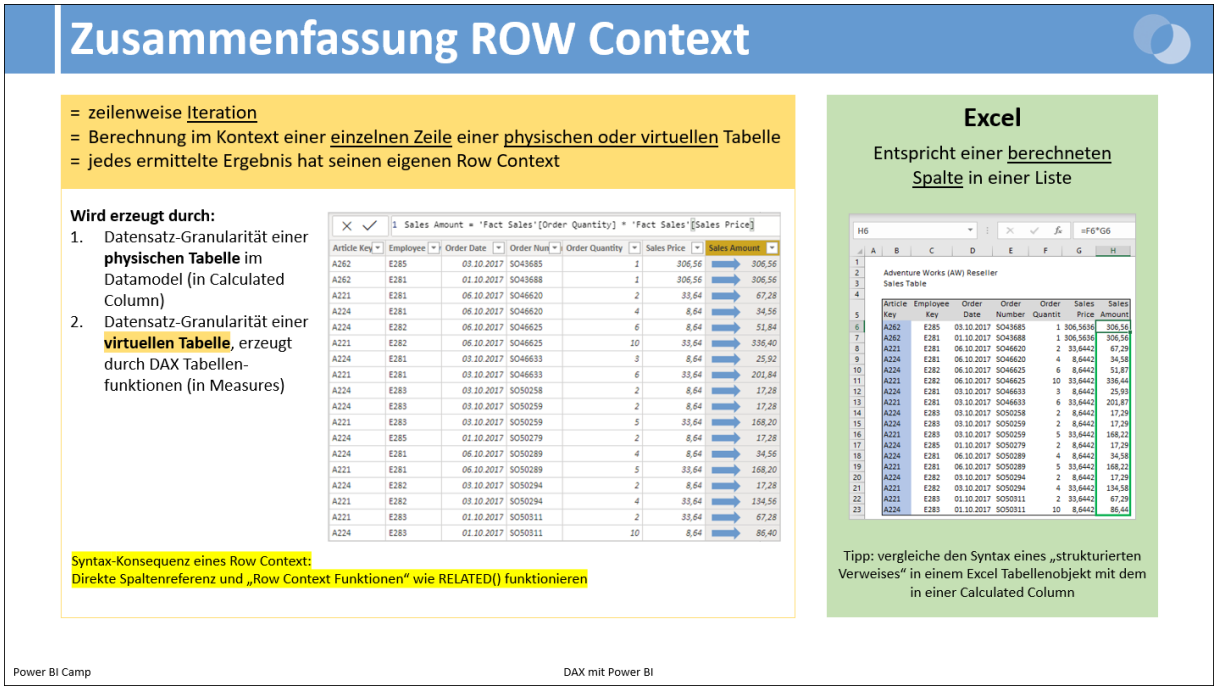
3.3.2 KONZEPT DES FILTER CONTEXT



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

Row Context heißt, 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):

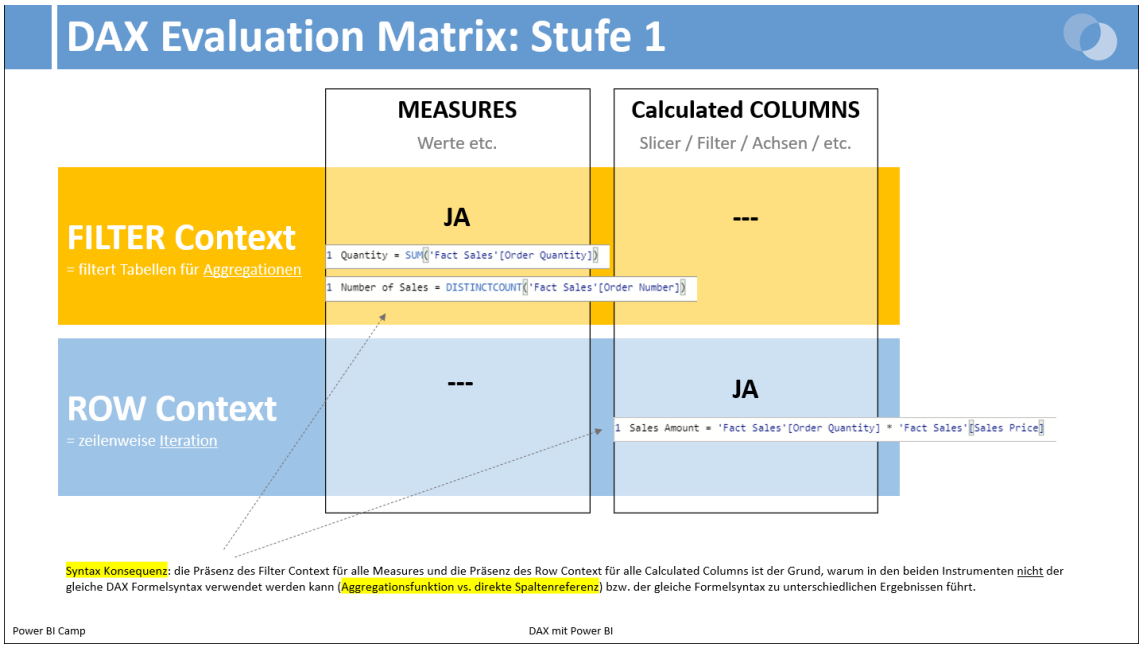


Tipp: vergleiche den Syntax eines „strukturierten Verweises" in einem Excel Tabellenobjekt mit dem in einer Calculated Column – diese sind fast ident!

3.3.4 EVALUATION MATRIX – STUFE 1 BIS 3

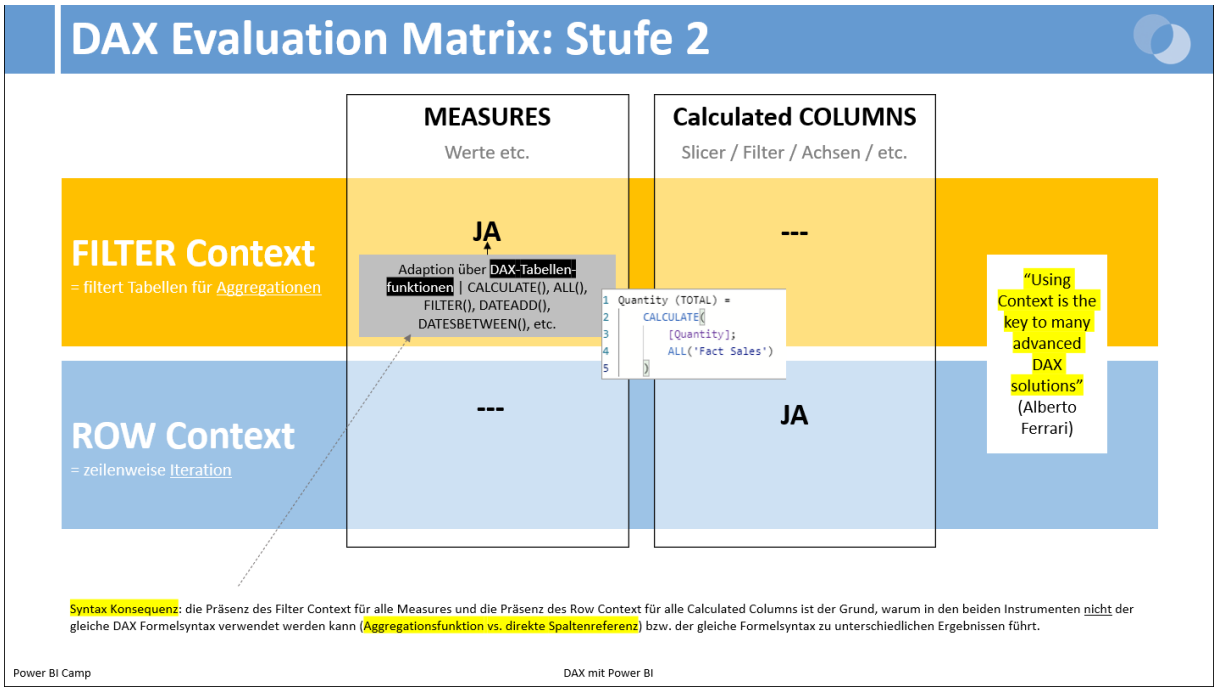
Grundsätzlich ist bis hierher alles sehr einfach:

- **Measures** arbeiten immer in einem Filter Context
- **Calculated Columns** arbeiten 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:

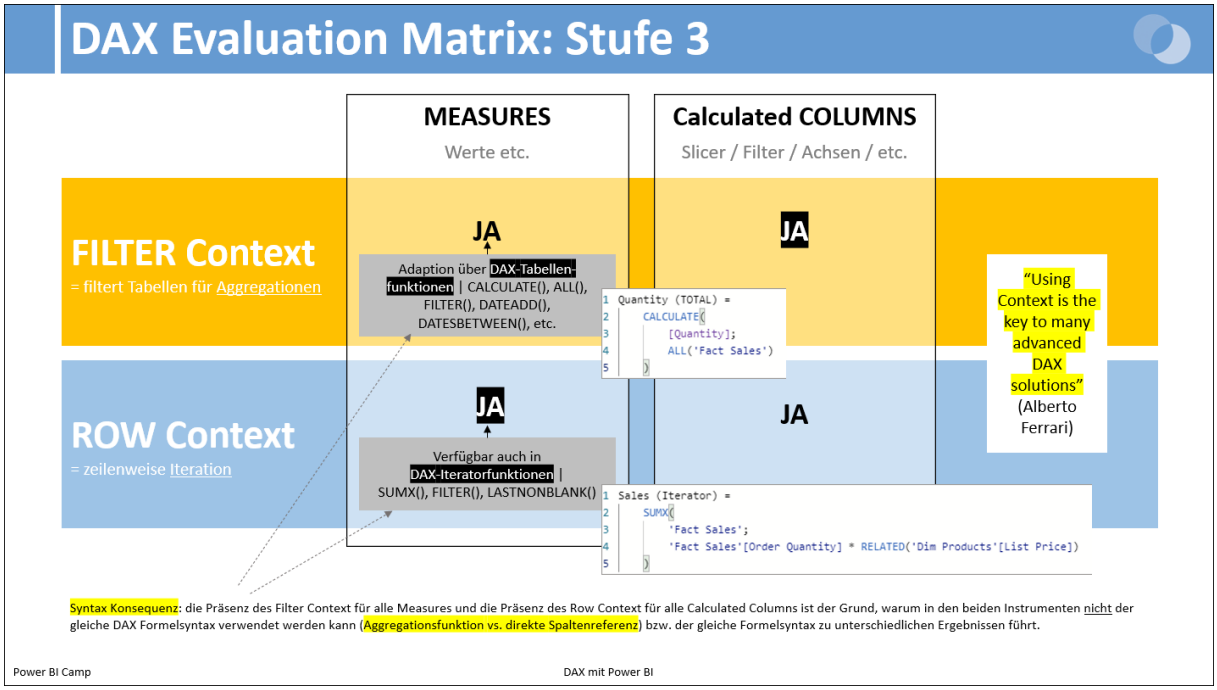


Wäre das alles, wären die Berechnungsmöglichkeiten in DAX sehr überschaubar.

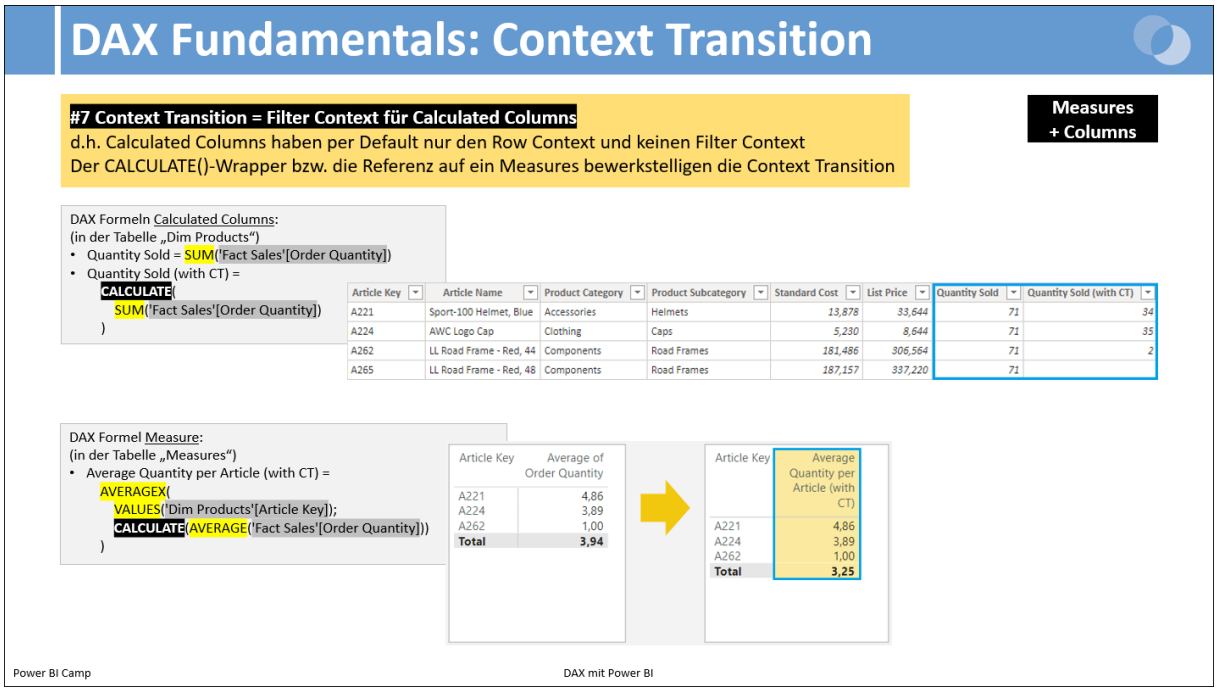
- Mit den DAX-Tabellenfunktionen – iVm einem CALCULATE() oder SUMX() – kann der Filter Context adaptiert werden und damit sehr mächtige Berechnungen wie Year-to-Date, Year-over-Year, %-von-Gesamt und vieles mehr bewerkstelligt werden.



- 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



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)

DAX Patterns: Verstehen der Context Transition

Anwendungsgebiete
1. **Calculated Column: Ermittlung der Absatzmenge aus Faktentabelle in der Dimensionstabelle „Dim Products"**

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

```
Quantity Sold (with CT) =
  CALCULATE(
    SUM('Fact Sales'[Order Quantity])
  )
```

DAX Patterns:
- C06.2 Lookup in verknüpfter Faktentabelle
- C08.1 Statische Segmentierung nach Neukunden
- C08.2 Statische Segmentierung nach Kleinkunden
- C08.3 Erster Timestamp innerhalb eines Process-Keys (als Column)
- M05.1 Gefilterte Werte (Syntaxvariante)
- M08.1 Bestandslogik (Syntaxvariante)
- M09.2 bis M09.5 Iterator Patterns
- M10 Ranking Patterns
- M17.3 Dynamische Altersberechnung

Article Key	Article Name	Product Categ	Product Subca	Standard C	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	Accessories	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 Patterns: Verstehen der Context Transition

Anwendungsgebiete:

2. Calculated Column: Ermittlung „Ist Bebucht ja/nein“

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"
  )
```

3. Measures mit Iteratorfunktionen

```
Average Quantity per Article (with CT) =
    AVERAGEX(
        VALUES('Dim Products'[Article Key]);
        CALCULATE(AVERAGE('Fact Sales'[Order Quantity]))
    )
```

Article Key	Quantity	Average of Order Quantity
A221	34	4,86
A224	35	3,89
A262	2	1,00
Total	71	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

DAX Patterns: Verstehen der Context Transition

Fortsetzung „Measures mit Iteratorfunktionen“
Iterator ohne CT liefert idR falsche Ergebnisse:

```
Average Quantity per Article (without CT) =
    AVERAGEX(
        VALUES('Dim Products'[Article Key]);
        AVERAGE('Fact Sales'[Order Quantity])
    )
```

Article Key	Average Quantity per Article (without CT)
A221	4,86
A224	3,89
A262	1,00
Total	3,94

```
Average Quantity per Article (without CT) =
    SUMX(
        VALUES('Dim Products'[Article Key]);
        AVERAGE('Fact Sales'[Order Quantity])
    )
```

Article Key	Average Quantity per Article (without CT)
A221	4,86
A224	3,89
A262	1,00
Total	15,78

Erklärung der Ergebnisse in der Abbildung oben:

→ Auf TOTAL-Ebene wird eine Tabelle mit allen 4 (nicht 3!) Artikel erzeugt mit jeweils dem ungefilterten Gesamt-Durchschnitt aller Faktendatensätze als Ergebnis je Datensatz. Diese 4 Werte werden dann über das AVERAGEX() verdurchschnittet - die Zahl bleibt daher unverändert der Gesamtdurchschnitt.

→ Auf Einzelzeilenebene fließt in den Iterator als Filter Context nur der jeweilige Artikel ein, es wird also eine virtuelle Tabelle mit nur 1 Zeile und dem Durchschnitt nur dieser Artikel aus der Faktentabelle ermittelt. Die Zahl ist daher auch ohne CT auf der Key Ebene richtig.

Die Variationen rechts unterlegen die beschriebene Berechnungsmethode.

```
Average Quantity per Article (without CT) =
    SUMX(
        VALUES('Fact Sales'[Article Key]);
        AVERAGE('Fact Sales'[Order Quantity])
    )
```

Article Key	Average Quantity per Article (without CT)
A221	4,86
A224	3,89
A262	1,00
Total	11,83

```
Average Quantity per Article (without CT) =
    SUMX(
        VALUES('Dim Products'[Article Key]);
        SUM('Fact Sales'[Order Quantity])
    )
```

Article Key	Average Quantity per Article (without CT)
A221	34
A224	35
A262	2
Total	284

Power BI Camp DAX mit Power BI

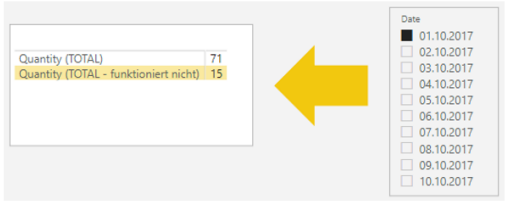
Weiterführend:
https://www.sqlbi.com/articles/understanding-context-transition/

3.3.6 KONZEPT DES OUTER FILTER CONTEXT UND DER SKALARE

DAX Fundamentals: DAX Variablen

#9.1 DAX Variablen ermöglichen einen Outer Filter Context
d.h. Variablen erzeugen bei der Ermittlung eines Skalarwertes eine Konstante
d.h. Variablen werden dort berechnet, wo sie definiert sind und nicht wo sie verwendet werden
d.h. der Outer Filter Context kommt zur Anwendung, wenn die Variable bspw. außerhalb des CALCULCATE() oder der SUMX() Funktion platziert wird

Measures



Alternative DAX Funktionen:
- EARLIER()
- EARLIEST()

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

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

Die gezeigte Syntax Variante liefert nicht das gewünschte Ergebnis, weil die ALL() Funktion im inneren Filter Context nicht mehr auf die Konstante wirken kann.

Power BI Camp DAX mit Power BI

DAX Fundamentals: DAX Variablen

#9.2 DAX Variablen konvertieren Skalare in Konstanten
d.h. in einem CALCULATE() dürfen – anders als in FILTER() – keine Skalare sondern nur Tabellen als Filterbedingung verwendet werden -> Variablen schaffen hier Abhilfe

Measures

```
Sales Annual (v1 Scalar) =
    CALCULATE(
        [Sales YTD];
        ALL('Dim Dates');
        'Dim Dates'[Date] <=
            CALCULATE(
                MAX('Fact Sales'[Order Date]);
                ALL('Fact Sales')
            )
    )
```
A function 'CALCULATE' has been used in a True/False expression that is used as a table filter expression. This is not allowed.

```
Sales Annual (v2 Scalar) =
    CALCULATE(
        [Sales YTD];
        ALL('Dim Dates');
        'Dim Dates'[Date] <= [Highest Date (absolute)]
    )
```
A function 'CALCULATE' has been used in a True/False expression that is used as a table filter expression. This is not allowed.

Date	Sales	Highest Date (absolute)	Sales Annual (v3 Scalar)	Sales Annual (v4 Scalar)
01.10.2017	477,58	06.10.2017	2.059,58	2.059,58
02.10.2017		06.10.2017	2.059,58	2.059,58
03.10.2017	889,02	06.10.2017	2.059,58	2.059,58
04.10.2017		06.10.2017	2.059,58	2.059,58
05.10.2017		06.10.2017	2.059,58	2.059,58
06.10.2017	692,97	06.10.2017	2.059,58	2.059,58
07.10.2017		06.10.2017	2.059,58	2.059,58
08.10.2017		06.10.2017	2.059,58	2.059,58
09.10.2017		06.10.2017	2.059,58	2.059,58
10.10.2017		06.10.2017	2.059,58	2.059,58
Total	**2.059,58**	**06.10.2017**	**2.059,58**	**2.059,58**

```
Highest Date (absolute) =
    CALCULATE(
        MAX('Fact Sales'[Order Date]);
        ALL('Fact Sales')
    )
```

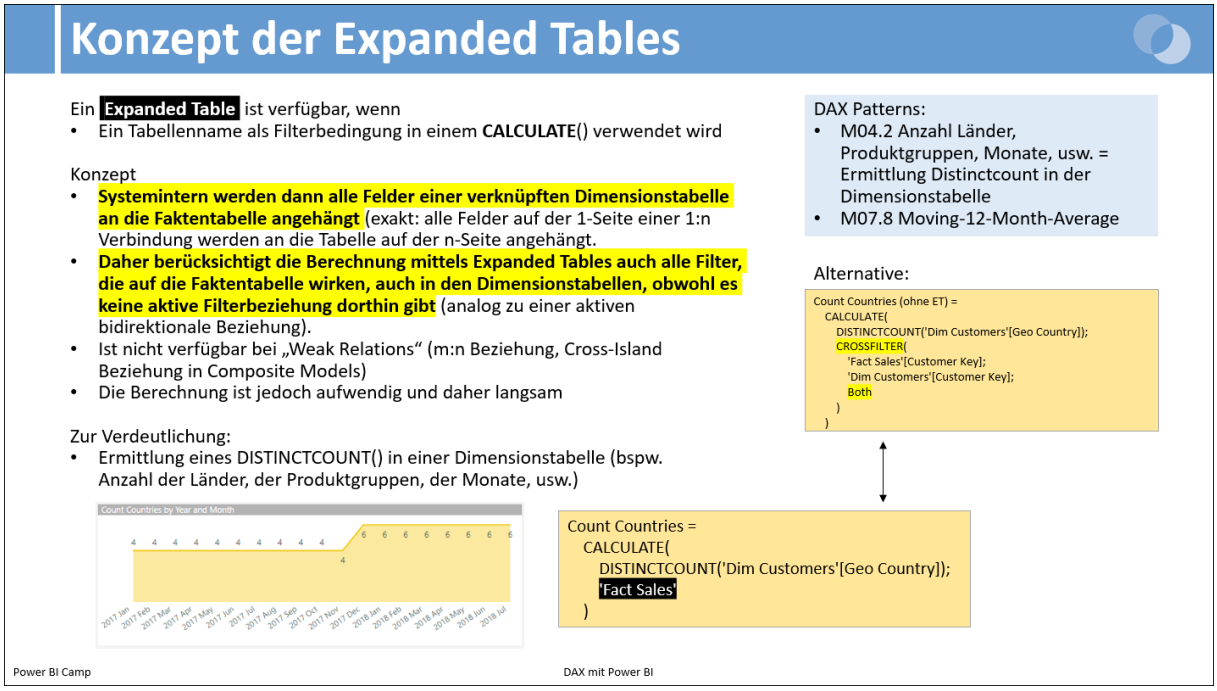
```
Sales Annual (v3 Scalar) =
VAR
    varHightestDate = [Highest Date (absolute)]
RETURN
    CALCULATE(
        [Sales YTD];
        ALL('Dim Dates');
        'Dim Dates'[Date] <= varHightestDate
    )
```

```
Sales Annual (v4 Scalar) =
    CALCULATE(
        [Sales YTD];
        FILTER(
            ALL('Dim Dates');
            'Dim Dates'[Date] <= [Highest Date (absolute)]
        )
    )
```

Power BI Camp DAX mit Power BI

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)

3.3.7 KONZEPT DER EXPANDED TABLES



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."

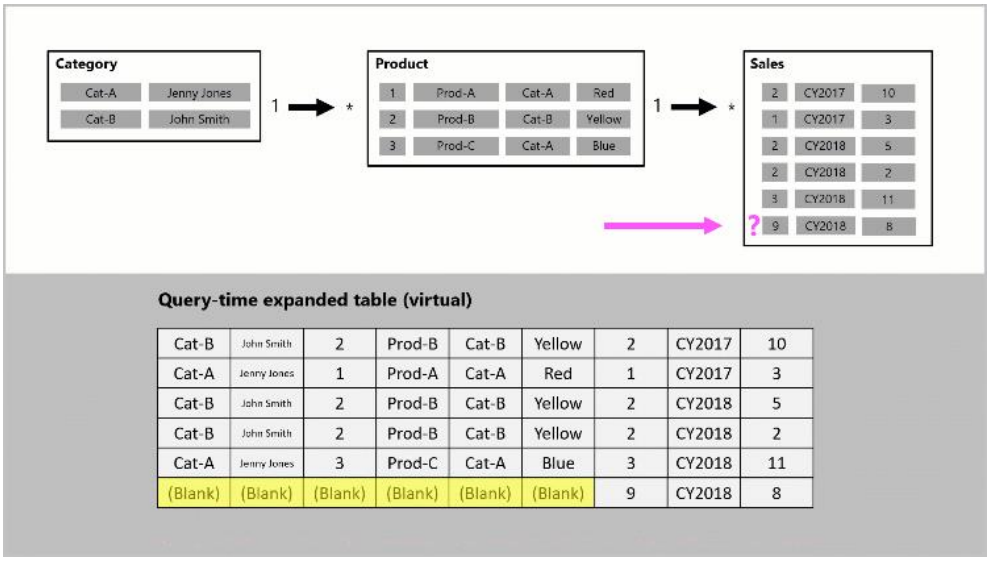
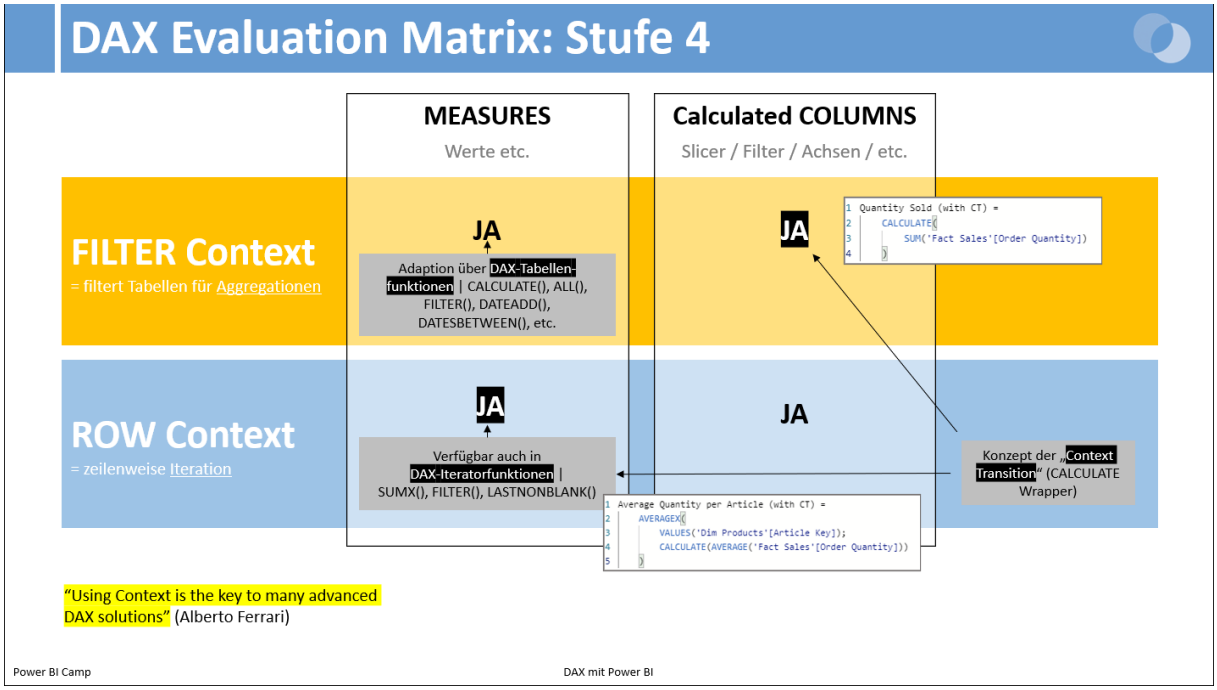


Abbildung: Expanded Table "Sales" mit Blankrow.

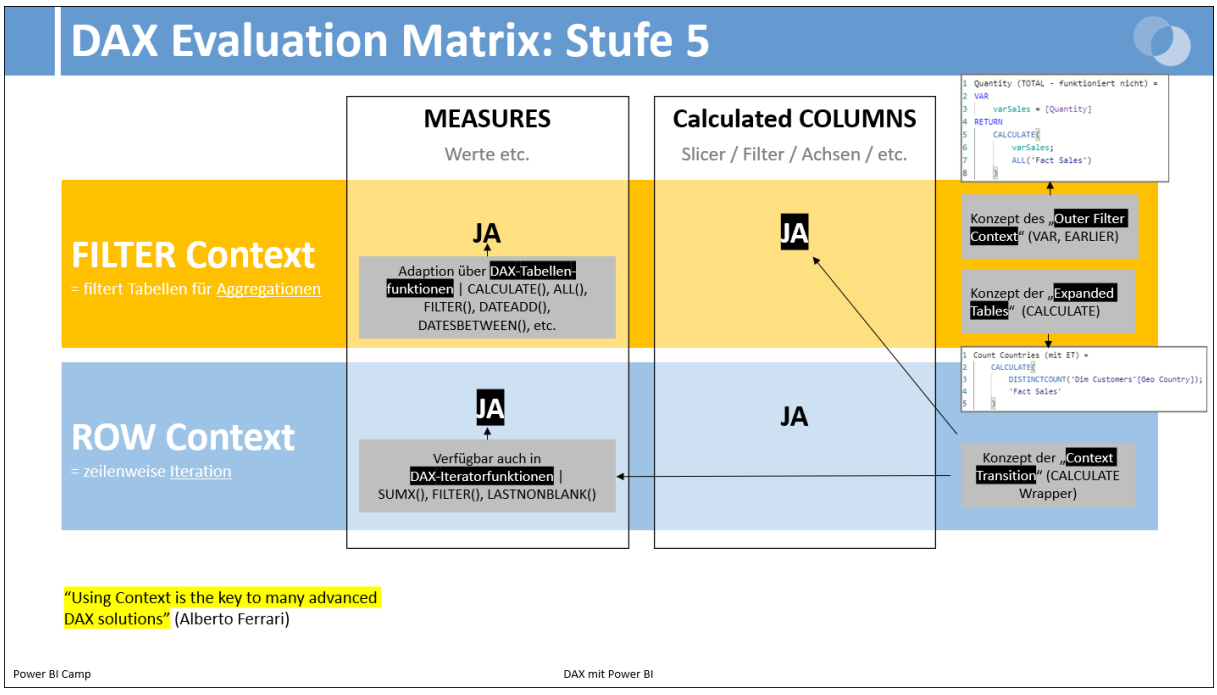
3.3.8 EVALUATION MATRIX – STUFE 4 BIS 6

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.

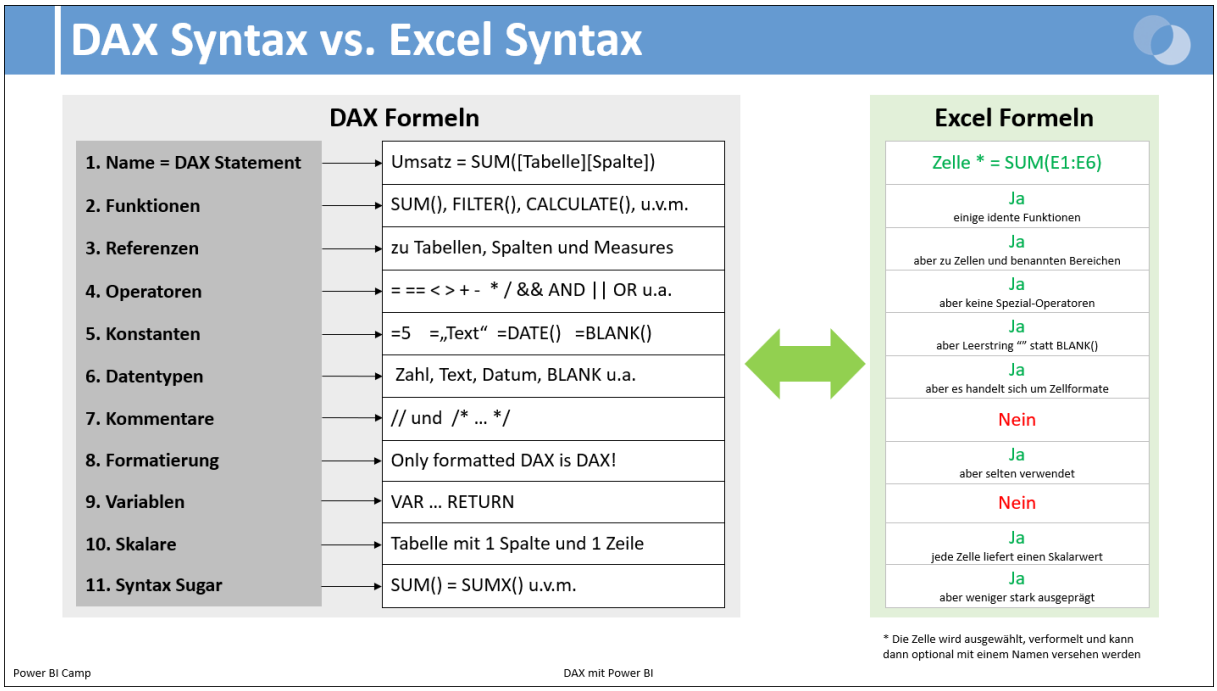


- Das Konzept des Outer Filter Context ermöglicht weitere Adaptionsmöglichkeiten des Filter Context.
- Das Konzept der Expanded Tables ermöglicht Berechnungen auf die verknüpften Dimensionstabellen auf Basis der Faktentabelle.



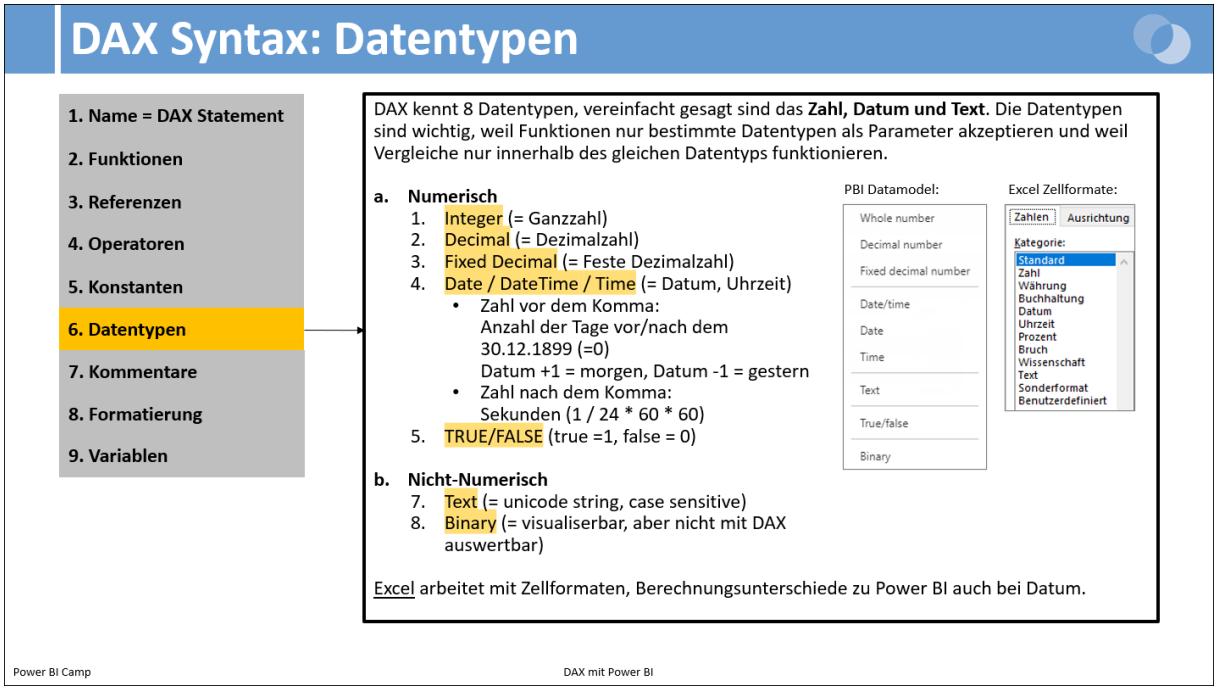
3.4 DAX SYNTAX

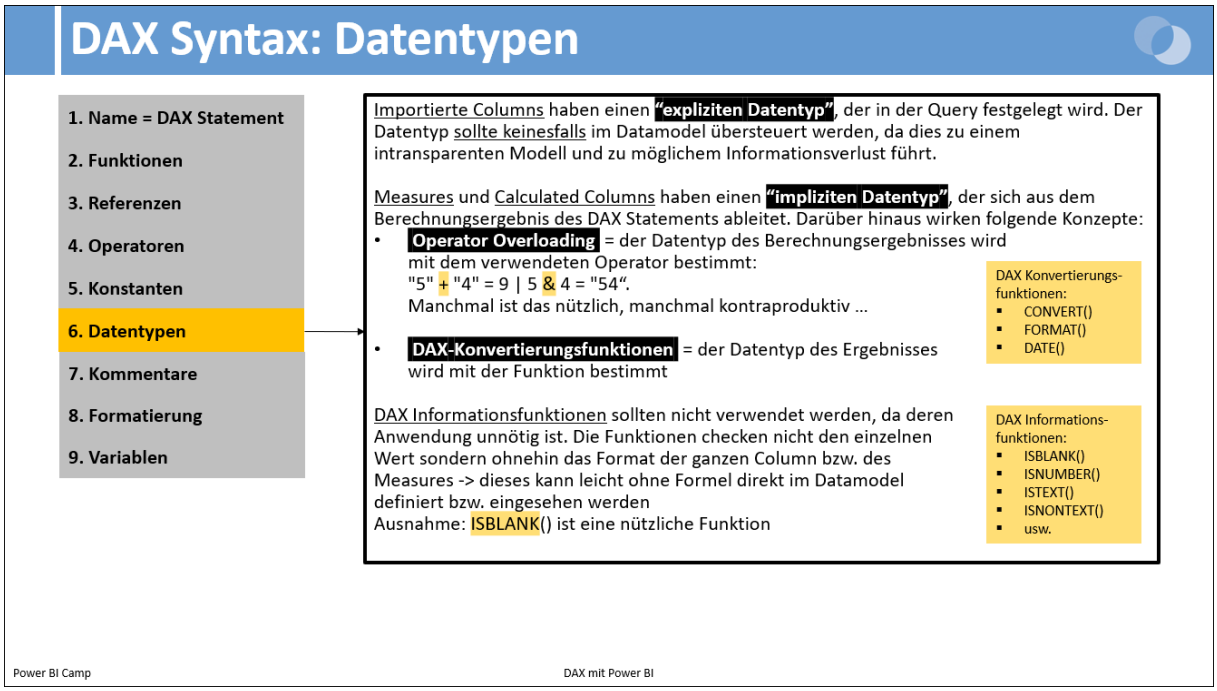
3.4.1 ALLGEMEINE DAX SYNTAX REGELN

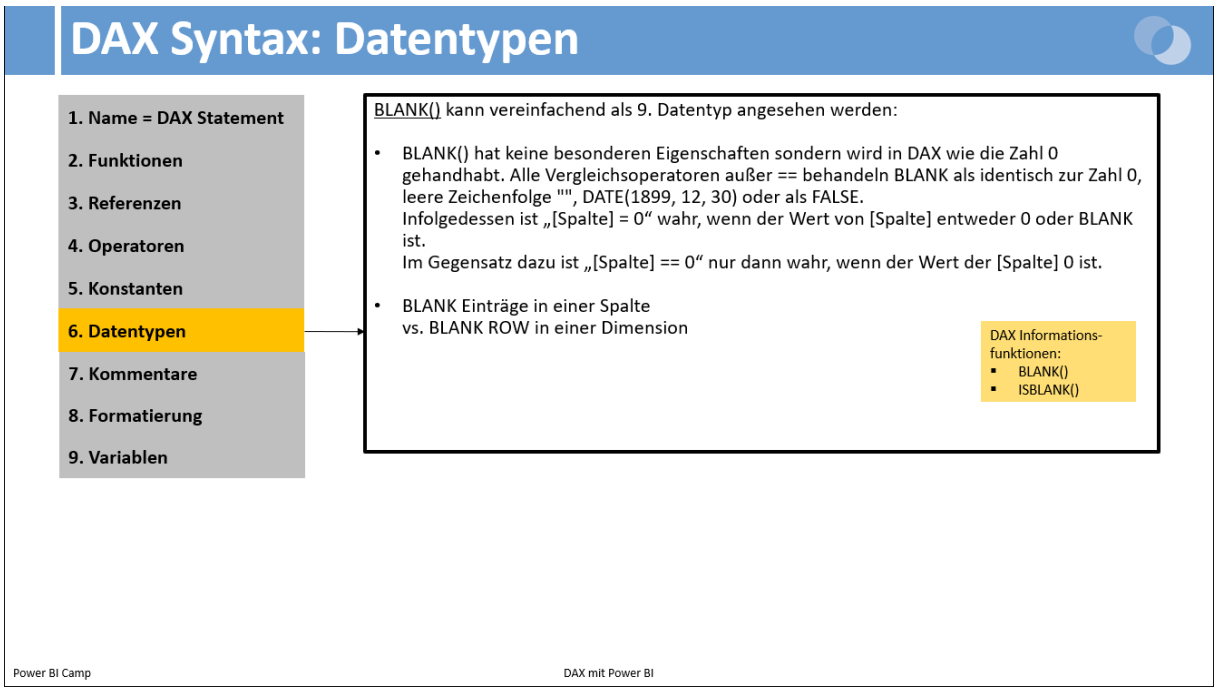


Weiterführend:
https://msdn.microsoft.com/en-us/query-bi/dax/dax-syntax-reference

3.4.2 DATENTYPEN IN DAX







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

3.4.3 SYNTAX SUGAR

Zahlreiche DAX Funktionen dienen lediglich dazu, einen anderen anspruchsvolleren Syntax zu vereinfachen.
Hier einige 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]
    ))
```

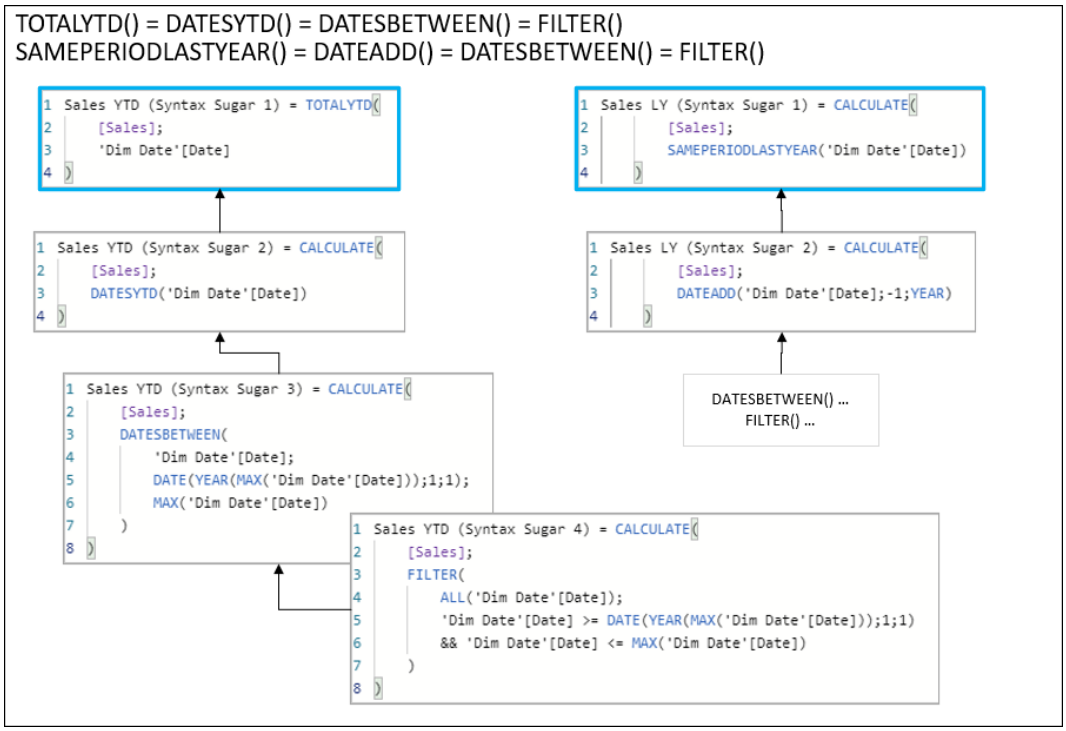
ISEMTPY(<Tabelle>) = “COUNTROWS(<Tabelle>) = 0”

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

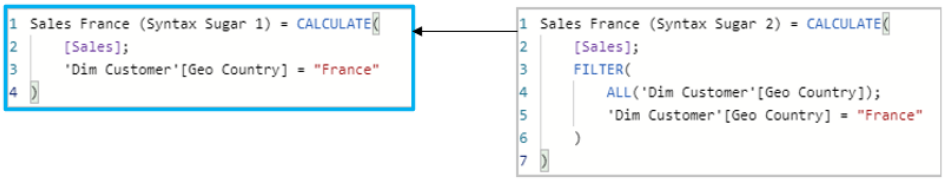
```
SWITCH (Syntax Sugar 1) =
    SWITCH(
        SELECTEDVALUE('Parameter: Scale'[Scale]);
        "in EUR";1;
        "in Tsd";0,001;
        "in Mio";0,000001;
        "please select a scale"
    )
```

```
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"
            )
        )
    )
```

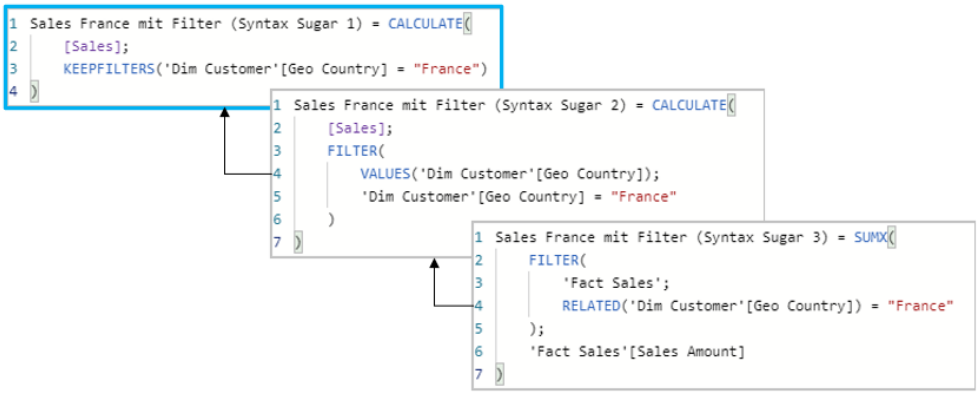
TOTALYTD() = DATESYTD() = DATESBETWEEN() = FILTER()
SAMEPERIODLASTYEAR() = DATEADD() = DATESBETWEEN() = FILTER()



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



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



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"
    )
```

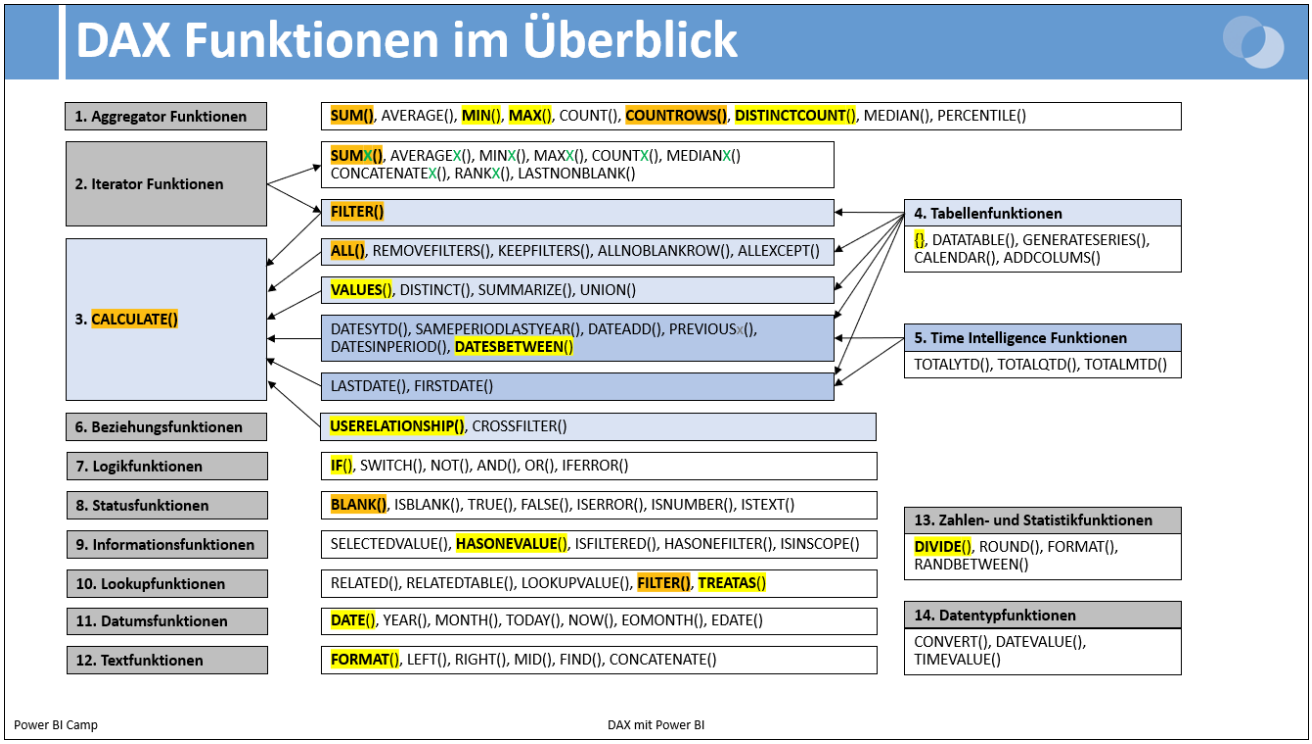
SELECTEDVALUE() = HASONEVALUE() mit VALUES() oder FIRSTNONBLANK() in der SWITCH() Funktion

```
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"
    )
```

3.5 ZENTRALE DAX FUNKTIONEN

Die – zahlreichen – DAX Funktionen können in die folgenden Gruppen eingeteilt werden, wichtig sind dabei auch die Querverbindungen beispielsweise bei den Tabellenfunktionen, von denen viele gleichzeitig auch Filterfunktionen sind:



Weiterführend:
https://docs.microsoft.com/de-de/dax/
https://dax.guide/