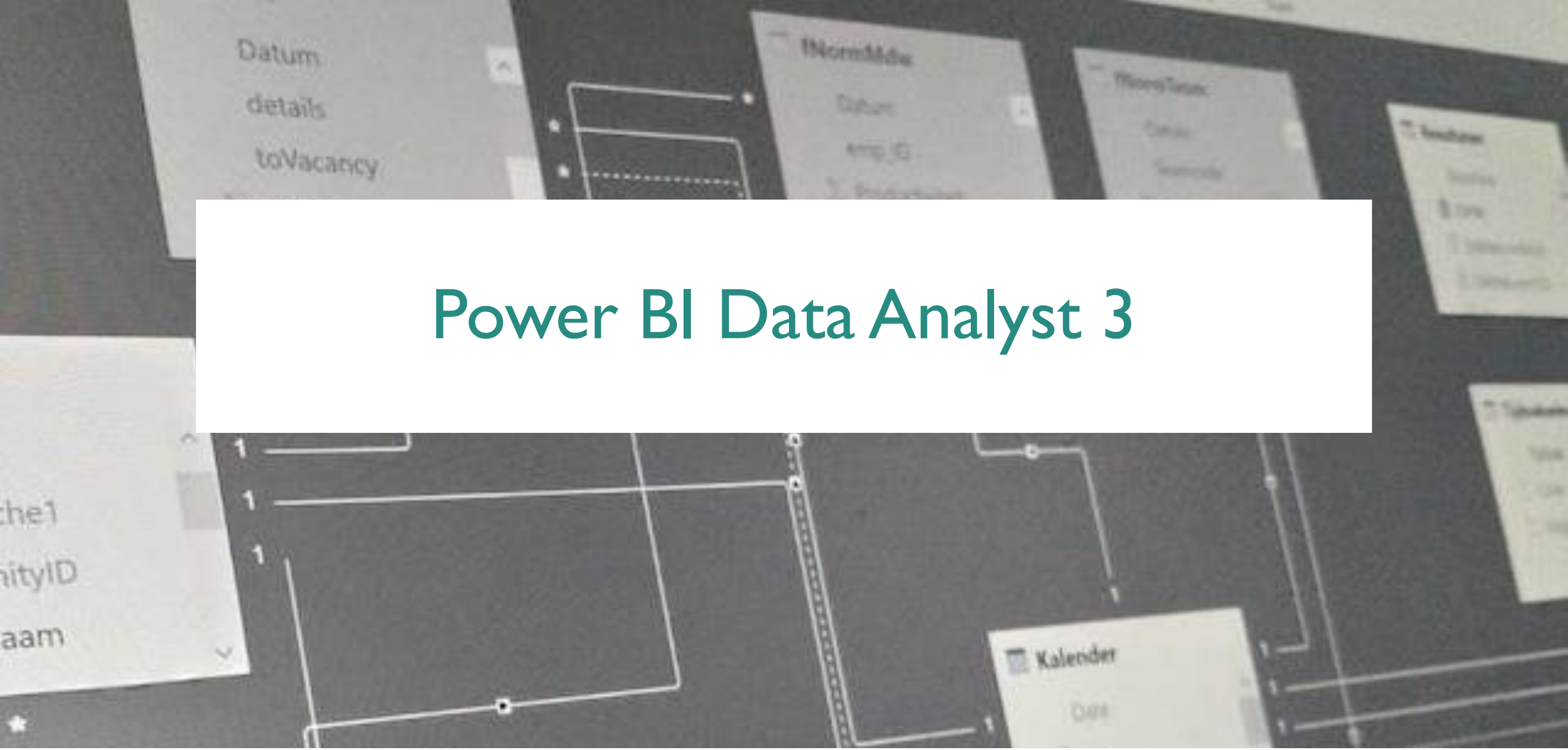


Power BI Data Analyst 3

Quanto
collective analytics

DAX basics

Calculated columns

- Add a column to a table, based on a calculation
- Is validated at time of creation or data refresh
- A calculated column is stored in the data model just like any other column

Example 1: =[UnitPrice]*[Number]

Example 2: =AVERAGE([Number])

Quanto
collective analytics

DAX syntax

- Calculated columns start with <name> =
- Refer to column names in brackets: [UnitPrice]
- Operators: + - * / &
- DAX functions: FUNCTION(Argument 1, Argument 2)
  - Function names are in English
  - Note the separator: , (or ; if you really want to)

Tips and guidelines

- Calculated columns must have format and other metadata
- Calculated columns can be used as the source or lookup for a relationship
- Think about naming your columns

Oh, and:

- Try to avoid using calculated columns as much as possible

Quanto
collective analytics

Measures (calculated fields)

- Add an aggregation to the model that can be used in output
- Are validated at time of use
- A measure does not take up space in the model (but does take processing time)

Example 1: Revenue = SUM(fSales[Amount])

DAX syntax (measures)

- Measures start with <Name> =
- Refer to column names in brackets with the table name: fSales[UnitPrice]
- A measure can be used as an argument (of the right type) for a DAX function
- Refer to measures in brackets: [Revenue] (without a table name)

Common DAX functions

Base aggregations:

- SUM()
- AVERAGE()
- MIN()
- MAX()
- COUNT()
- COUNTA()
- DISTINCTCOUNT()
- PRODUCT()

Other functions:

- BLANK()
- IF()
- DIVIDE()

Quanto
collective analytics

Tips and guidelines

- Use a calculation table (or measure table)
- Keep formulas short, and reuse whenever possible
- Measures can be hidden: use this for intermediate calculations
- Think about naming of measures
- Provide format for measures

Context

Quanto
collective analytics

Context

- Calculations in DAX are dynamic
- The results depend on the context in which the calculation is performed
- You can do almost anything with DAX by using and transforming context

- Types of context: row context, query context, filter context

Row context

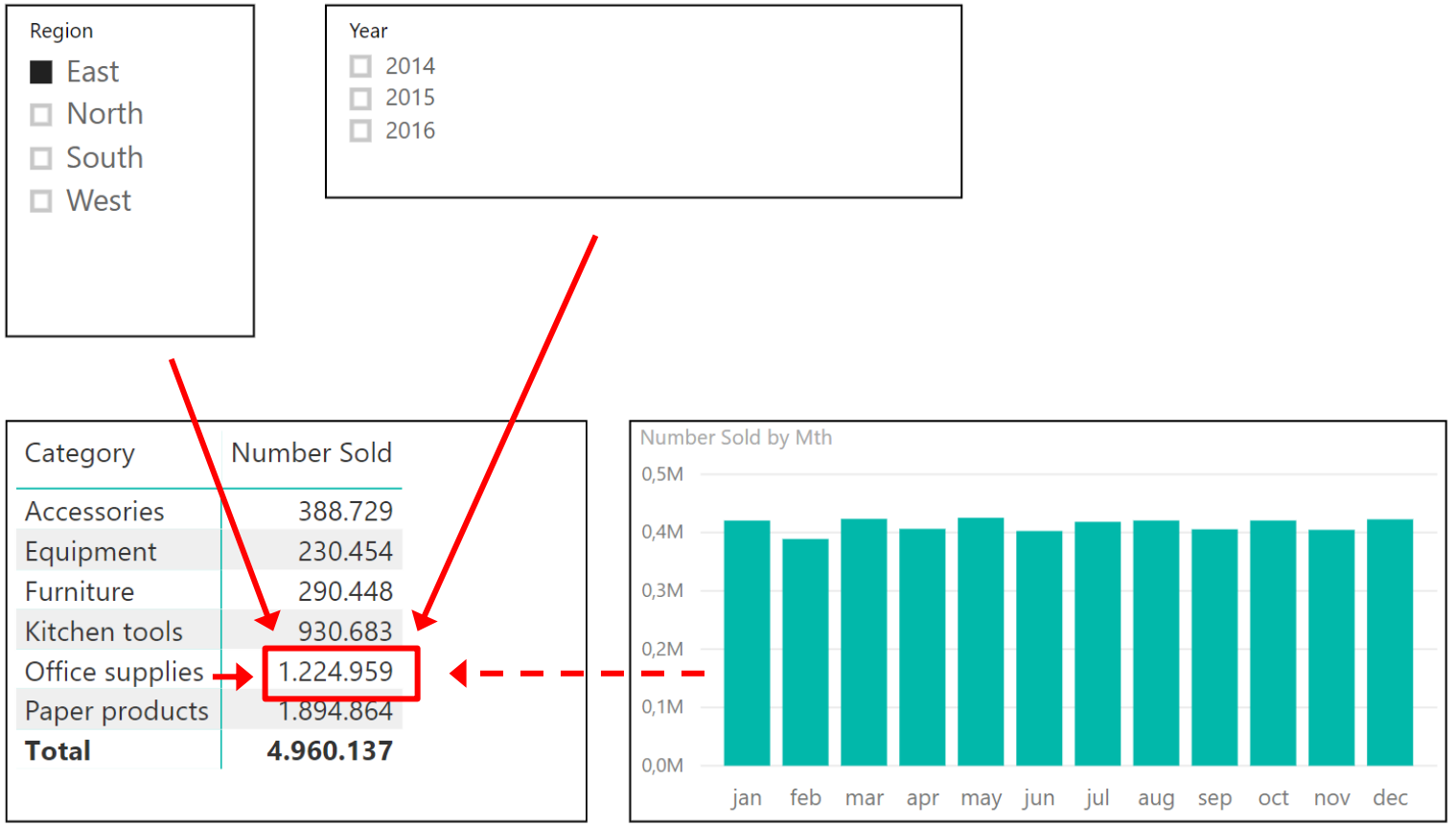
- Row context is the context in which a calculated column is computed (by row)
- In each row the model knows the value of each column
  - So you can directly reference a column: [UnitPrice]
- Values in other rows are not available

OrderDate	Product	UnitPrice	Number	Amount
13-10-2015	Pen	1,25	50	
25-11-2015	Pencil	0,95	150	
5-12-2015	Pencil	0,90	200	
8-12-2015	Notebook	2,45	80	
12-12-2015	Pen	1,30	50	

Quanto
collective analytics

Query context

Query context is the combination of *selections* and *filters* in the report that determine the result of a measure

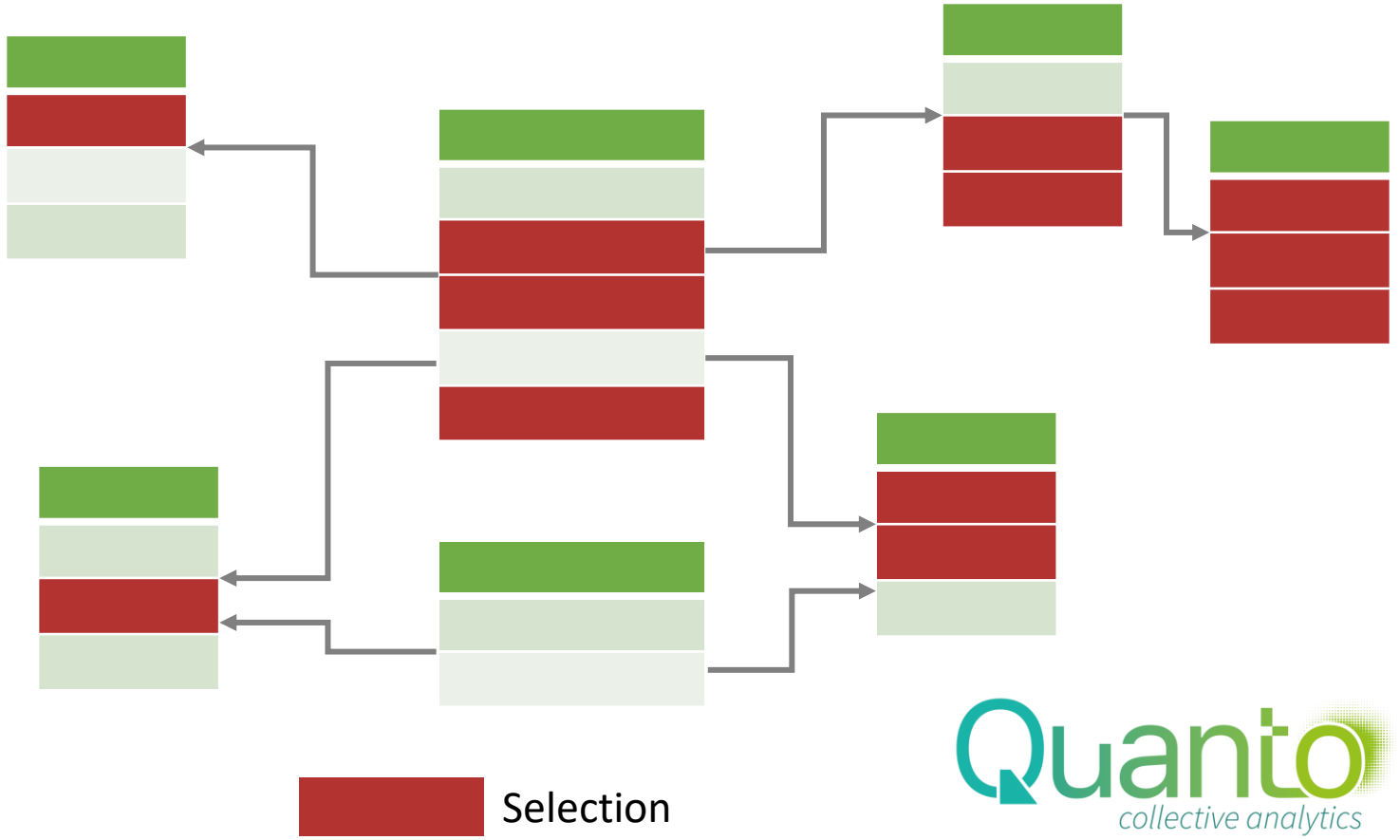


Quanto
collective analytics

Query context: selection

Selection in the report (slicers, page filters, row- or axis labels) correspond to rows in filter tables being selected

Relationships propagate selection resulting in selection of a number of rows in the fact tables



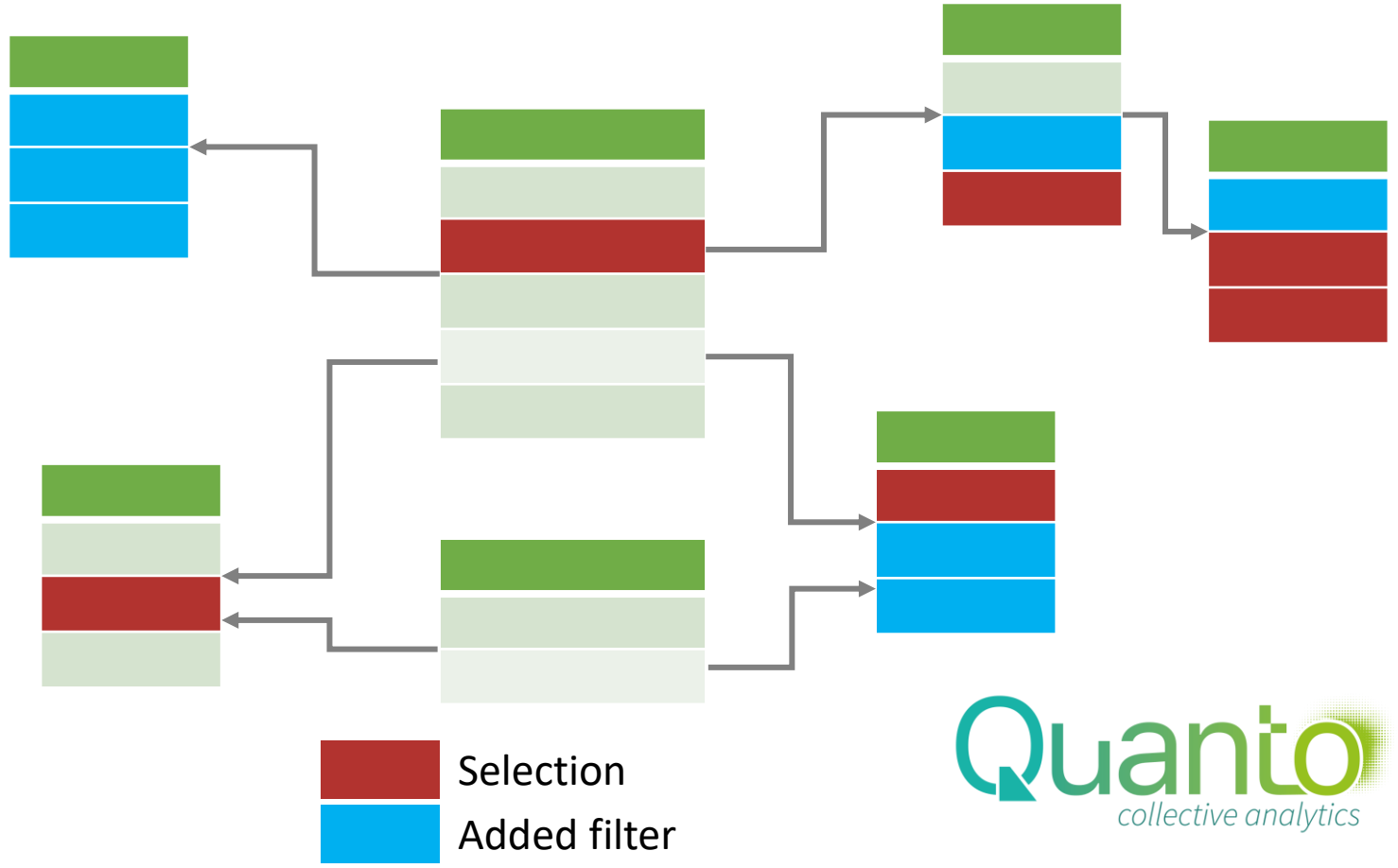
Quanto
collective analytics

Filter context

Quanto
collective analytics

Filter context

- Filter context is a context transformed with DAX
- Filter context behaves like query context
- But: you cannot derive filter context from what you see in the report



Quanto
collective analytics

Filtering with DAX: CALCULATE()

CALCULATE(<expression>; filter1; filter2; …)

SalesWest = CALCULATE([Sales], Customer[Region]="West")

CALCULATE does four things:

1. Create a filter context
2. Remove existing filters from columns or tables that are referenced in the filter arguments: Customer[Region]
3. Add new filters: Customer[Region]="West"
4. Compute <expression> in the new filter context

Quanto
collective analytics

The secret to success in Power BI

When you want your model to produce some results, start with wondering:

1. In which (query) context do I request this result
2. Which (filter) context will deliver the result
3. How to transform the query context into the filter context

ALL functions

These functions remove existing filters:

- ALL(Table[Column]) removes a filter from Table[Column]
  - Works with multiple columns as well
- ALL(Table) removes all filters from Table (all columns)
- ALLEXCEPT(Table;Table[Column]) removes all filters on columns in Table, except those on Table[Column]
  - Works with multiple columns as well

Some special filter functions

USERELATIONSHIP()

- Is used to activate an inactive relationship
- E.g. CALCULATE([Revenue]; USERELATIONSHIP(fSales[Payment date]; Date[Date]))

KEEPFILTERS()

- This changes the behavior of CALCULATE: instead of *replacing* existing filters, filters in the CALCULATE statement are *added* to the existing context
- E.g. CALCULATE(<expression>; KEEPFILTERS(<filter>))

Quanto
collective analytics

Some special filter functions (2)

CROSSFILTER()

- Is used to change the filter propagation behaviour of a relationship
- CROSSFILTER(Table1[Column1]; Table2[Column2]; <Mode>)
  - Both: relationship uses two-way crossfiltering
  - None: relationship uses no crossfiltering
  - OneWay: relationship uses default, one-way crossfiltering
- E.g. CALCULATE([Revenue]; CROSSFILTER(fSales[Payment date]; Date[Date]; Both))
- Mostly used to enable bi-directional crossfiltering, but can be used as an alternative to ALL() without loosing the context on the filter table

Quanto
collective analytics

Time Intelligence functions

Quanto
collective analytics

This used to be my favorite DAX function ☺

```
RevenueYTD = TOTALYTD([Revenue], Date[Date])
```

Quanto
collective analytics

Time intelligence functions

- Allow for intelligent filtering over time
- Prerequisite: the Date or Calendar table
  - One row per day
  - Other columns for filtering
  - All fact tables relating to the Date table
  - Use TRUNC([Date]) when needed
  - Use the ‘Mark as Date table’ feature
  - Date[Date] is an argument of every time intelligence function

Quanto
collective analytics

What TOTALYTD() does

TOTALYTD() is a shortcut for a YTD filter:

```
TOTALYTD([Revenue]; Date[Date])
```

is equivalent to

```
CALCULATE([Revenue]; DATESYTD(Date[Date]))
```

DATESYTD replace the current context on the Date table by the YTD period (from 1st day of the year to last day in the current context)

Note that TOTALYTD() can be used with any calculation, not only SUM(), e.g.

```
TOTALYTD([AvgPrice]; Date[Date]))
```

returns the year-to-date average price

Quanto
collective analytics

The most-used Time intelligence filters

- SAMEPERIODLASTYEAR() – returns the existing context, but one year earlier
- DATEADD() – returns the current context, shifted forward or backward by a number of periods
  - E.g. DATEADD(Date[Date];-2;quarter)
- DATESINPERIOD() – returns a period extending from a start date to a number of intervals earlier or later
  - E.g. DATESINPERIOD(Date[Date];TODAY();-6;month)
- DATESBETWEEN() – returns a period between two dates
  - E.g. DATESBETWEEN(Date[Date];[Start_Date];[End_Date])

Quanto
collective analytics