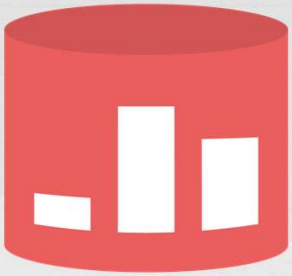
www.sqlbi.com

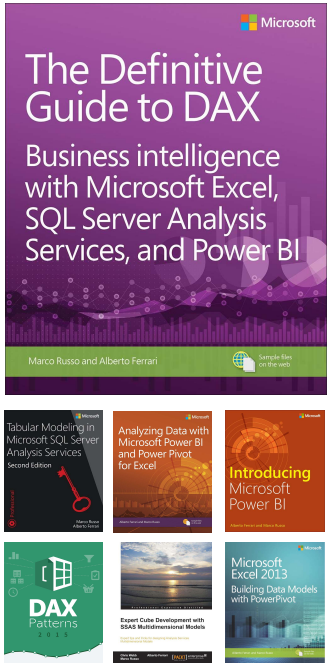


sqlbi

DAX Best Practices

Marco Russo
marco@sqlbi.com

We write
Books



We teach
Courses

MASTERING DAX Workshop

OPTIMIZING DAX Workshop

DATA MODELING For Power BI

POWER BI DASHBOARD DESIGN COURSE

SSAS TABULAR WORKSHOP

We provide
Consulting

- Remote Consulting
- Power BI/SSAS Optimization
- BI Architectural Review
- On-Site Consulting
- Custom Training & Mentoring

We are recognized
BI Experts

Microsoft Partner
Gold Business Intelligence
Gold Data Platform

Microsoft CERTIFIED Technology Specialist

Microsoft CERTIFIED IT Professional

Microsoft CERTIFIED Trainer

MVP Microsoft Most Valuable Professional

SSAS MAESTRO

sqlbi
www.sqlbi.com

DAX Best Practices

- Naming convention
- Use variables
- Errors are good
- FILTER vs. CALCULATE and complex filter conditions in calculated columns
- Beware of bidirectional filters
- Avoid context transition large iterators

Naming convention

- Never use table names for measure references
  - Use [Total Sales] instead of Sales[Total Sales]
  - Measure reference: [measure]

- Always use table names for column references
  - Use Sales[Quantity] instead of [Quantity]
  - Even in the definition of a calculated column referencing columns in the same table
  - Column reference: 'table'[column]

- https://www.sqlbi.com/articles/rules-for-dax-code-formatting/
- https://www.sqlbi.com/articles/duplicated-names-in-dax/

sqlbi

Demo

Naming Convention

Use variables everywhere

- Improve code readability
  - Reduce nested calls
  - Make the code more «procedural» and less «functional»
  - Remove EARLIER
- Improve performance
  - Variables are evaluated only once
  - Avoid double evaluation of the same expression

https://www.sqlbi.com/articles/variables-in-dax/
https://www.sqlbi.com/articles/dax-coding-style-using-variables/

sqlbi

Demo

Use variables

Improve code readability

```
TaxedSales :=
SUMX (
    Sales,
    Sales[Quantity] * Sales[Unit Price] * ( 1 + Sales[Tax Percentage] )
)
```

sqlbi

Improve code readability

```
TaxedSalesExplained :=
SUMX (
    ADDCOLUMNS (
        ADDCOLUMNS (
            ADDCOLUMNS (
                Sales,
                "LineAmount", Sales[Quantity] * Sales[Unit Price]
            ),
            "Taxes", [LineAmount] * Sales[Tax Percentage]
        ),
        "TaxedAmount", [LineAmount] + [Taxes]
    ),
    [TaxedAmount]
)
```

sqlbi

Improve code readability

```
TaxedSalesVariables :=
SUMX (
    Sales,
    VAR LineAmount = Sales[Quantity] * Sales[Unit Price]
    VAR Taxes = LineAmount * Sales[Tax Percentage]
    VAR TaxedAmount = LineAmount + Taxes
    RETURN
        TaxedAmount
)
```

sqlbi

Improve performance

```
Average Positive Balances :=
AVERAGEX (
    Customer,
    IF (
        [Balance] > 0,
        [Balance]
    )
)
```

sqlbi

Improve performance

```
Average Positive Balances :=
AVERAGEX (
    Customer,
    VAR CustomerBalance = [Balance]
    RETURN
        IF (
            CustomerBalance > 0,
            CustomerBalance
        )
)
```

sqlbi

Errors are good

- Show an error when the calculation is not safe
  - A report that is not working makes people angry
  - A report providing wrong numbers will get you fired
- Do not use IFERROR
  - Hiding errors could be a bad idea
- Use ERROR to provide better error description
  - Constant string, cannot be dynamic
  - Should make it easier to understand where is the issue in data

Demo

Errors are good

Avoid IFERROR

```
Estimated Quantity :=
IFERROR (
    [Sales Amount] / VALUES ( Product[Unit Price] ),
    BLANK ()
)
```

sqlbi

Avoid IFERROR

```
Estimated Quantity :=
DIVIDE (
    [Sales Amount],
    SELECTEDVALUE ( Product[Unit Price] )
)
```

Avoid IFERROR

```
Estimated Quantity :=
IF (
    COUNTROWS ( VALUES ( Product[Unit Price] ) ) > 1,
    ERROR ( "More than one Unit Price selected" ),
    IF (
        VALUES ( Product[Unit Price] ) = 0,
        ERROR ( "Unit Price is zero or not available" ),
        [Sales Amount] / VALUES ( Product[Unit Price] )
    )
)
```

sqlbi

Avoid IFERROR

```
Estimated Quantity :=
VAR UnitPrice = VALUES ( Product[Unit Price] )
RETURN IF (
    COUNTROWS ( UnitPrice ) > 1,
    ERROR ( "More than one Unit Price selected" ),
    IF (
        UnitPrice = 0,
        ERROR ( "Unit Price is zero or not available" ),
        [Sales Amount] / UnitPrice
    )
)
```

sqlbi

Demo

Complex filter conditions

Complex filter conditions

- Apply filters using CALCULATE/SUM instead of SUMX/FILTER
  - DAX can optimize simple conditions
  - Complex conditions requires some iterator
- Use column filters rather than table filters
  - Use ALL with multiple columns for same table OR conditions
  - Use CROSSJOIN with multiple columns for different tables OR conditions
  - Use KEEPFILTERS rather than VALUES
- Consider calculated column to persist the result of complex condition
  - The condition must be static and not dynamic
  - The result of the condition should have a low granularity (e.g. true/false)

Demo

Beware of bidirectional filters

Beware of bidirectional filters

- Use bidirectional filter only when necessary
  - Use CROSSFILTER instead of bidirectional filter whenever possible
  - Only use bidirectional filter in the data model when fact tables are reacheable only on one side of the relationship
- Unexpected behavior with bidirectional filter
  - Paths using single direction filters wins over bidirectional filters when ambiguity is involved
  - USERELATIONSHIP can create ambiguity that would not be accepted in the data model. In this case, the ambiguity is solved by applying the shortest path rule

Demo

Context transition

Avoid context transition in large iterators

- Context transition is expensive
  - Materialization of uncompressed data required
  - One of the common source of performance issues
- Context transition could provide unexpected result
  - Tables without primary key will aggregate identical rows
  - Classical error: aggregation of context transition produces inflated results
- Avoid context transition in iterators or tables without primary key

Recap

- Naming convention
- Use variables
- Errors are good
- FILTER vs. CALCULATE and complex filter conditions in calculated columns
- Beware of bidirectional filters
- Avoid context transition large iterators

sqlbi

Just like Jimi Hendrix …

We love to get feedback

Please complete the session feedback forms

sqlbi

SQLBits - It's all about the community...

Please visit Community Corner, we are trying this year to get more people to learn about the SQL Community, equally if you would be happy to visit the community corner we'd really appreciate it.

sqlbi

Thank you!

Check our articles, whitepapers and courses on

www.sqlbi.com