Note DAX

**SCALAR Vs. TABULAR FUNCTIONS**

1. Hàm vô hướng

- Hàm vô hướng là hàm trả về một giá trị duy nhất, có thể là date, numberic, text, vv

- SUM/Average/Min/Max

- SUMX/AverageX/CountX/MinX/MaxX

- LastDate/FirstDate

- CALCULATE

- Có thể sử dung các hàm này trong Measure or Calculated Column

- Nếu sử dụng trong Measure => kết quả trong visual

- Nếu sử dụng trong Calculated Column => kết quả trong mỗi dòng của bảng

VD: Sales Amount = Calculate(SUM(FactSales[SalesAmount])

- Sử dụng CALCULATE với FILTER như 1 đối số khi cần:

+ Muốn xem các giá trị thỏa mãn điều kiện bộ lọc các giá trị khác trả về BLANK

+ So sánh giá trị cột với giá trị cột, giá trị cột vs measure, giá trị measure vs measure.

+ Bộ lọc có nhiều điều kiện OR

+ Mọi ngôn ngữ tính toán đều tính từ trong ra ngoài có nghĩa là tính toán với hàm FILTER đầu tiên trả về kết quả là một bảng đã được lọc rồi mới sử dụng hàm CALCULATE

<https://stackoverflow.com/questions/72000696/why-calculate-is-not-modifying-filter-context-when-used-with-filter>

- Sử dụng KEEPFILTER như 1 đối số thay vì FILTER để bảo tồn filter đang tồn tại. Điều này giúp giữ lại các bộ lọc đã được áp dụng trước đó và bổ sung thêm các bộ lọc mới mà không ghi đè lên filter context hiện có.

- Trong DAX, các biến được định nghĩa trong khối VAR và được đánh giá theo thứ tự xuất hiện trước khi đến câu lệnh RETURN. Bạn không thể thay đổi giá trị của một biến đã được định nghĩa trước đó bằng cách sử dụng CALCULATE hoặc bất kỳ hàm nào khác trong câu lệnh RETURN.

2. Hàm dạng bảng

- Hàm tabular trả về kết quả là 1 bảng không phải giá trị duy nhất

- Bảng có thể có nhiều cột hoặc chỉ có 1 cột phụ thuộc vào hàm được sử dụng nhưng nó sẽ có dạng bảng với nhiều giá trị bên trong

VD: Copy of Customer Table = ALL(Dimcustomer)

ALL returns a table as the result

- ALL, ALLEXCEPT

- RelatedTable

- SamePeriodLastYear

- DatesBetween/ DatesInPeriod

- Summarize/GroupBy

- TreatAs

…

- Sử dụng chủ yếu trong calculated table

3. Sử dụng sai các các hàm

a. Scalar function để tạo calculated table

- Scalar func cannot be the main output of the calculated table cuz a function that returns one single value, cannot build a table

b. Tabular function trả về kết quả là 1 Measure hoặc calculated column

- return a table and measure expects a single value

**What can you DO? Sử dụng các hàm lồng nhau**

* Muốn sử dụng tabular func trong Measures: k sử dụng trực tiếp được tuy nhiên có thể sử dụng như kết quả trung gian được sử dụng bên trong hàm vô hướng

VD: Same Period Last Year = CALCULATE( SUM(FactInternetSales[SalesAmount]), SAMEPERIODLASTYEAR( FactInternetSales[OrderDate].[Date] ) )

* Muốn sử dụng scalar func trong Calculated table: k sử dụng trực tiếp được tuy nhiên có thể sử dụng như kết quả trung gian được sử bên trong hàm tabular.

VD: GroupBy - with aggregation = GROUPBY( DimCustomer,DimCustomer[EnglishEducation], "Row Count", COUNTX( CURRENTGROUP(), DimCustomer[CustomerKey] ) )

4. Làm sao để biết được khi nào sử dụng hàm nào là hợp lí? A screenshot of a computer

Description automatically generated

- Sự giới hạn: Calculate trả về 1 giá trị vô hướng nhưng k thể sử dụng trong GroupBy func

- Ngoại lệ: có những hàm kh phải hàm scalar cũng k phải hàm tabular, nó k trả về kết quả. Ví dụ, hàm CrossFilter dùng để thay đổi trạng thái relationship và chỉ được sử dụng bên trong calculate func

**AGGREGATION FUNCTIONS**

* Với tập dữ liệu lớn (1M+ dòng) sử dụng COUNTROWS và VALUES có thể ít gây áp lực lên DAX engine hơn so với việc sử dụng DISTINCTCOUNT
* SWITCH(True) là mẫu phổ biến thay thế cho hàm lồng IF
* COALESCE thay thế IF và ISBANK

**ADVANCED CALCULATE**

* Theo mặc định, calculated columns chỉ hiểu row context chứ không hiểu filter context. Để tạo filter context ở row-level => dùng CALCULATE
* Khi bạn sử dụng các hàm như SUM, SUMX, hoặc CALCULATE, DAX tự động tạo ra filter context dựa trên các bộ lọc hiện tại trong báo cáo hoặc mô hình dữ liệu.
* Bất kỳ khi nào bạn viết hàm có sử dụng các biểu thức logic (IN, >,<,=,…)

= > tạo bảng nội bộ . Bảng nội bộ này được xử lý bằng các hàm như FILTER và ALL để áp dụng filter context tương ứng

Ví dụ :

Giả sử bạn muốn tính tổng doanh thu cho sản phẩm có giá trị SalesAmount > 1000

HighSales =

CALCULATE(

SUM(Sales[SalesAmount]),

Sales[SalesAmount] > 1000

)

Trong ví dụ này biểu thức Sales[SalesAmount] >1000 tạo ra 1 bảng nội bọ chứa các hàng trong bảng Sales có cột SalesAmount >1000. Sau đó CALCULATE áp dụng filter context để tính tổng

* Hàm ALL xóa bỏ tất cả các bộ lọc trên một cột hoặc bảng trả về tất cả các giá trị trong cột hoặc bảng mà không áp dụng bất kì bộ lọc nào.
* Thứ tự đánh giá: Khi CALCULATE gặp ALL đầu tiên nó sẽ xóa hết các bộ lọc hiện có sau đó các bộ lọc khác trong câu lệnh CALCULATE sẽ được áp dụng .
* Các bộ lọc thông thường của CALCULATE:

+ ALL xóa bỏ tất cả các bộ lọc trên một cột hoặc bảng cụ thể

+ ALLEXCEPT xóa bỏ tất các các bộ lọc ngoại trừ các bộ lọc trên các côt được chỉ định

+ ALLSELECTED: Xóa bỏ tất cả các bộ lọc trên một cột hoặc bảng, nhưng giữ lại các bộ lọc được áp dụng bởi các lựa chọn người dùng trên giao diện.

+ VALUES trả về một bảng gồm các giá trị duy nhất trong 1 cột

+ FILTER trả về 1 bảng đã được lọc dựa trên điều kiện nhất định

+ REMOVEFILTERS loại bỏ tất cả các bộ lọc khỏi một cột hoặc bảng cụ thể, k trả về 1 bảng.

+ KEEPFILTERS giữ nguyên các bộ lọc hiện có và thêm bộ lọc mới mà không ghi đè các bộ lọc hiện tại

+ USERELATIONSHIP chỉ định mối quan hệ giữa các bảng nên được sử dụng trong 1 phép tính hoặc để ghi đè mối quan hệ dữ liệu mặc định trong mô hình.

+ CROSSFILTER dc sử dụng để chỉ định cách áp dụng mối quan hệ giữa các bảng

CROSSFILTER(<table1[colun1]>, <table2[column2]>, <cross\_filter\_type> )

* CALCULATE đánh giá tất cả các đối số bộ lọc như một bảng

**TABLE AND FILTER FUNCTIONS**

* Filter Data: ALL, FILTER, DISTINCT, VALUES, ALLEXCEPT, ALLSELECTED

|  |  |
| --- | --- |
| ALL | Vừa là bộ lọc bảng vừa là như một bộ điều chỉnh cho hàm CALCULATE.  Không chấp nhận biểu thức bảng, chỉ chấp nhận tham chiếu đến bảng vật lý hoặc các cột. |
| FILTER | Vừa là một hàm bảng tạo ra các bảng mới dựa trên điều kiện lọc, vừa là một hàm lặp, lặp qua từng hàng của bảng để áp dụng điều kiện lọc |
| DISTINCT | Trả về một bảng hoặc cột chứa các giá trị duy nhất từ cột được chỉ định.  Các giá trị trùng lặp loại bỏ, k show blank row |
| VALUES | Khi mà đối số đầu vào là một column name, VALUES trả về một bảng có 1 cột bao gồm các giá trị distinct của cột được chỉ định. Giá trị BLANK có thể được thêm vào nếu có giá trị null trong cột.  Khi đối số đầu vào là table name, trả về các dòng từ bảng được chỉ định. Các dòng duplicate được giữ nguyên.  Một hàng BLANK có thể được thêm vào nếu có giá trị NULL trong bảng.  VALUES luôn hiển thị BLANK row nếu có các giá trị null trong bảng tra cứu => kiểm tra bảng có giá trị thiếu hoặc trùng lặp hay không. |
| ALLEXCEPT(Tablename,Columname, Columnname,…) |  **Loại bỏ tất cả các bộ lọc** từ bảng được chỉ định ngoại trừ các bộ lọc trên các cột được liệt kê.   **Giữ lại các bộ lọc** chỉ trên các cột được liệt kê trong danh sách tham số.  TableName phải là bảng vật lí. ColumnName có thể cùng bảng tham chiếu hoặc hoặc nằm ở phía one-side của mối quan hệ (Expanded Table). |
| ALLSELECTED | Được sử dụng để loại bỏ tất cả các bộ lọc hiện tại từ bảng hoặc cột, nhưng vẫn giữ lại các bộ lọc đã được chọn (selected) |
| SELECTEDVALUE | Trả về giá trị khi chỉ có một giá trị được chọn trong cột, nếu không, trả về kết quả thay thế tùy chọn.  **Ứng dụng**: Truy xuất giá trị cột từ cùng một bảng, hiển thị chi tiết mà không hiển thị tổng phụ hoặc tổng cộng, tạo tiêu đề báo cáo động, kiểm tra điều kiện trước khi tính toán. |

**The differences**

+ ALLSELECTED: removes only filters applied to the specified columns in the query

+ ALLEXCEPT: removes all filters except the filters applied to the specified columns in the query

* Add Data: SELECTEDCOLUMNS, ADDCOLUMNS, SUMMARIZE, GROUPBY

|  |  |
| --- | --- |
| SELECTCOLUMNS(<Table>, [<Name>], <Expression>, <Name>], …) | Trả về một bảng với các cột được chọn từ bảng cộng với bất kì cột mới được chỉ định bởi biểu thức DAX.  Nó có thể chấp nhận bảng ảo (FILTER, ALL)  SELECTCOLUMNS bắt đầu với một bảng trống và chỉ thêm các cột được chỉ định vào bảng mới. |
| ADDCOLUMNS(<table>, <name>, <expression>[, <name>, <expression>]…) | Hàm ADDCOLUMNS thêm các cột mới vào bảng hiện có bằng cách tính toán giá trị của các cột mới này từ các biểu thức DAX. Giữ nguyên các cột ban đầu  ADDCOLUMNS bắt đầu với toàn bộ bảng ban đầu và thêm các cột mới vào bảng đó. |
| SUMMARIZE(<table>, <groupBy\_columnName>[, <groupBy\_columnName>]…[, <name>, <expression>]… | Sử dụng giống SELECT DISTINCT + GROUP BY trong SQL.  Các cột chỉ định sẽ được ưu tiên từ trái qua phải tức là nếu chúng ta đặt cột chỉ định nào trước thì sẽ ưu tiên nhóm theo cột đó trước.  Chỉ thao tác với bảng vật lý |
| GROUPBY(<table> [,<groupBy\_columnName>[, <groupBy\_columnName> [, …]]] [,  <name>, <expression> [, <name>, <expression> [, …]]]) | Hàm GROUPBY là hàm tạo ra một bảng mà được nhóm lại theo cột chỉ định.  Thao tác với bảng vật lý và bảng ảo.  **<expression>**: Biểu thức DAX để tính giá trị cho cột mới (thường là CURRENTGROUP). |

* Create Data: ROW, DATATABLE, GENERATESERIES , { } Table

|  |  |
| --- | --- |
| ROW(<column1>, <expression1>, [<column2>, <expression2>]...) | Hàm ROW tạo một bảng đơn lẻ chứa một hàng duy nhất. Nó hữu ích để tạo ra các bản ghi tạm thời hoặc khi bạn cần kết hợp các giá trị vào một bảng duy nhất |
| DATATABLE(  <column1>, <datatype1>,  <column2>, <datatype2>,  ...  {  {<value1>, <value2>, ...},  {<value1>, <value2>, ...},  ...  }  ) | Hàm DATATABLE cho phép bạn tạo một bảng với nhiều hàng và cột một cách rõ ràng bằng cách xác định kiểu dữ liệu và giá trị cho từng cột |
| GENERATESERIES(<start>, <end>, [<increment>]) | Tạo một chuỗi số nguyên, thường được sử dụng để tạo ra các chuỗi ngày hoặc số |
| { <value1>, <value2>, ... } | Cú pháp { } được sử dụng để tạo ra các bảng tạm thời hoặc các danh sách giá trị đơn giản, tương tự như mảng. |

**CALCULATED TABLE JOINS**

* **CROSSJOIN** trả về một bảng chứa tích Đề các của các bảng được chỉ định. Điều này có nghĩa là mỗi hàng của bảng đầu tiên được kết hợp với mỗi hàng của bảng thứ hai tạo ra một bảng với tất cả các kết hợp có thể

Ví dụ 1: Bảng A có *a* dòng *b* cột, bảng B có *c* dòng và *d* cột

+ Số lượng hàng = a\*c

+ Số lượng cột = b+d

+ Tên cột: các cột có tên khác nhau để tránh xung đột

+ Loại bảng: Bảng vật lí hoặc biểu thức bảng như VALUES, DISTINCT,

FILTER,…

Ví dụ 2:

CrossJoinedTable = CROSSJOIN(

VALUES(Product[ProductName]),

DISTINCT(Sales[Region]),

VALUES(Date[CalendarYear])

)

Trong ví dụ này, CROSSJOIN tạo ra một bảng CrossJoinedTable chứa tất cả các kết hợp có thể giữa các giá trị duy nhất của cột ProductName, Region, và CalendarYear.

* **UNION**  kết hợp các dòng từ hai hoặc nhiều hơn các bảng có cấu trúc cột như nhau. Các bảng này có thể là bảng vật lý hoặc bảng ảo được tạo từ DAX như DATATABLE.
* **EXCEPT(**LeftTable, RightTable) trả về tất cả các hàng từ bảng bên trái nhưng không có ở bảng bên phải.

+ LeftTable phải là bảng vật lý có trong mô hình dữ liệu

+ Cả hai bảng phải có cùng số cột

+ Tên các cột được xác định bởi bảng trái

+ Bảng kết quả không giữ lại mối quan hệ với các bảng khác

* **INTERSECT** trả về tất các dòng từ bảng bên trái và cũng xuất hiện ở bảng bên phải.

+ LeftTable là bảng vật lý

+ Thứ tự của các bảng quan trọng. (T1, T2) có thể cho kết quả khác so với (T2, T1).

+ Bản sao các dòng được duy trì

+ Tên cột được xác định vởi bảng bên trái

+ Các cột được so sánh dựa trên vị trí trong các bảng tương ứng của chúng

* Sự khác biệt giữa CALCULATE và CALCULATETABLE

+ CALCULATE trả về một giá trị vô hướng (for measures)

+ CALCULATETABLE trả về một bảng ảo (for table), có thể được sử dụng như một đối số cho các hàm bảng khác hoặc là một bảng con trong biểu thức DAX

**RELATIONSHIP FUNCTIONS**

* **RELATED:** Đang ở phía "many", muốn lấy dữ liệu từ phía "one" của mối quan hệ

Ví dụ :

TotalSales = SUMX( Orders, RELATED(Customers[CustomerName]) )

Giả sử bạn có mối quan hệ giữa bảng Orders (phía "nhiều") và Customers (phía "một"), trong đó Orders có một cột CustomerID là khóa ngoại đến Customers.

* **RELATETABLE:** Đang ở phía "one", muốn lấy dữ liệu từ phía "many" của mối quan hệ

+ Nó được sử dụng thường xuyên với các hàm tổng hợp như COUNTROWS, SUMX, AVERAGEX,... để tính toán các chỉ số (measures) từ các bảng liên quan.

+ RELATEDTABLE là một hàm thuận tiện (shortcut) để tạo bảng được tính toán (CALCULATEDTABLE) mà không cần các biểu thức logic phức tạp. Nó thực hiện một chuyển đổi ngữ cảnh từ ngữ cảnh hàng (row context) sang ngữ cảnh bộ lọc (filter context), để chỉ trả về các hàng thỏa mãn các điều kiện bộ lọc.

* **USERELATIONSHIP** (ColumnName1=Foreignkey,ColumnName2=Primarykey) được sử dụng để kích hoạt một mối quan hệ không hoạt động giữa hai bảng trong mô hình dữ liệu của bạn. Mối quan hệ này sẽ chỉ được áp dụng trong ngữ cảnh của các hàm mà yêu cầu tham số bộ lọc, chẳng hạn như CALCULATE hay TOTALYTD.
* **CROSSFILTER**( LeftColumn = “Many” side, RightColumn = “One” side, CrossFilterType) hoạt động giống như USERELATIONSHIP nhưng thay đổi cách thức làm việc của bộ lọc giữa hai cột của hai bảng khác nhau. Ví dụ: Sửa đổi mối quan hệ cross filter direction (thành one hoặc both) hoặc vô hiệu hóa truyền bộ lọc.
* **TREATAS** cho phép tạo mqh ảo từ một hoặc nhiều cột trong bảng ảo đến một column trong bảng thực

**ITERATOR FUNCTIONS**

* Iterator cardinality là số dòng trong các bảng được lặp lại, càng nhiều dòng thì cardinality càng cao

+ Physical relationships: cardinality = max number of unique rows in the largest table

+ Virtual relationships: cardinality = number of unique rows in each table multiplied together

* CONCATENATEX: lặp qua từng hàng trong <table>, áp dụng <expression> cho mỗi hàng để trích xuất giá trị, sau đó nối chúng lại thành một chuỗi kết quả.
* **SUMX**: Tính tổng của một biểu thức tính toán cho từng hàng trong một bảng.
* **AVERAGEX**: Tính trung bình của một biểu thức tính toán cho từng hàng trong một bảng.
* **MINX và MAXX**: Tìm giá trị nhỏ nhất và lớn nhất của một biểu thức tính toán cho từng hàng trong một bảng.
* **COUNTROWS**: Đếm số hàng trong một bảng hoặc kết quả của một hàm iterator khác.
* **FILTER**: Lọc các hàng từ một bảng dựa trên một điều kiện được chỉ định.
* **AVERAGE và AVERAGEX bỏ qua blank cell vì vậy để dánh giá một giá trị trung bình qua 1 khoảng thời gian gồm cả những ngày không có doanh thu sử dụng DIVIDE và COUNTROWS thay thế.**
* **Moving Average using variables, FILTES và AVERAGEX (**<https://medium.com/analytics-vidhya/moving-average-using-dax-power-bi-413a31099091>**)**
* Moving Average using DATESINPERIOD & AVERAGEX (<https://www.sqlbi.com/articles/rolling-12-months-average-in-dax>)
* DATESINPERIOD làm việc giống hàm `DATEDIFF` (<https://dax.guide/datesinperiod/>)
* **Các Đặc Điểm Của Hàm Iterator**

+ Lặp qua từng hàng: Hàm iterator lặp qua từng hàng trong bảng được chỉ định và áp dụng biểu thức tính toán lên từng hàng đó.

+ Áp dụng biểu thức: Biểu thức tính toán có thể là một biểu thức đơn giản như một cột trong bảng hoặc một biểu thức phức tạp hơn.

+ Phạm vi áp dụng: Hàm iterator thường được sử dụng trong các công thức CALCULATE, CALCULATETABLE, hoặc trong các biểu thức tính toán cho các cột tính toán (calculated columns) và các biểu thức đo lường (measures)

**ADVANCED TIME INTELLIGENCE**

* CALENĐAR làm việc giống GENERATSERIES nhưng datatype là DATETIME
* CALENDARAUTO(Fiscal\_Year\_End\_Month) : nếu không có đối số sẽ tự động bắt đầu vào ngày Jan 01.