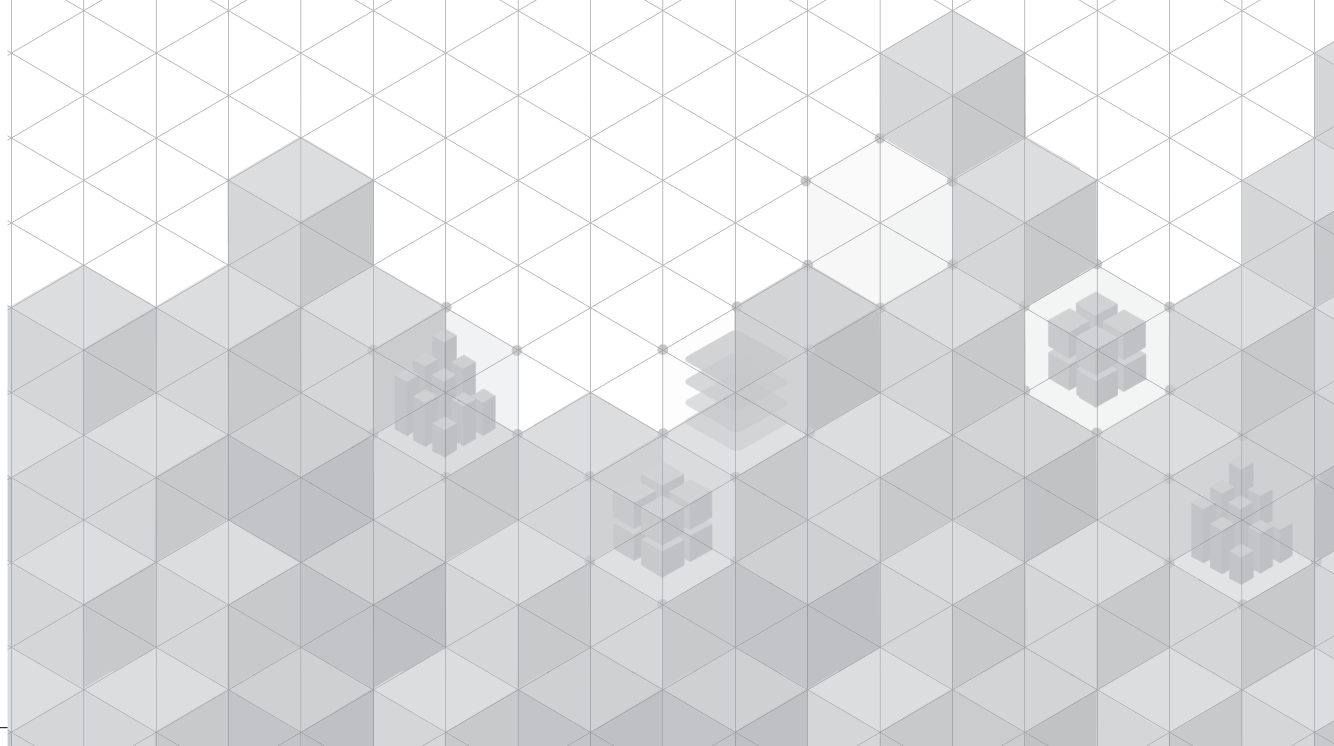


SQL Server® 2012  
商業智慧2.0

## 第08章DAX資料分析 語言進階計算

---

進階DAX函數  
DAX陳述句查詢



在上一章中，介紹了DAX函數的基本介紹，同時從商務分析人員所熟悉的Excel函數與資訊技術人員所熟悉的T-SQL函數來比對，說明DAX函數的功能。在本章中，我們將要介紹更進階的DAX技術議題，包含DAX函數中最特殊的篩選函數與路徑函數；另外也會說明如何運用DAX來進行資料查詢，並整理了常見的商業分析主題，好讓各位了解如何運用DAX來實做這些商業分析。

## 08-01 進階DAX函數

### 8-1-1 篩選函數

篩選函數是DAX中最為複雜的一環。多維度分析的精神在於預先彙總，因此MDX的重點就在於如何定義分析範圍，以調閱出預先彙總的結果。但因表格式模型是將資料壓縮存放於記憶體中，也就是需要什麼樣的資訊都需要透過記憶體即時計算。因此，篩選函數的重要性就在於協助定義分析的範圍，以方便計算引擎根據此範圍算出答案。

在T-SQL語法裡，是透過Where或者是Having的句型來定義篩選範圍。但DAX函數最大的特色在於，它只保留最純粹的函數結構（沒有其他查詢修飾詞、句型結構），所以即使要做出如同T-SQL中子查詢的效果，在DAX中還是會以純函數結構來處理，也就是說，篩選函數的引數中包括了資料表（或資料行），而回傳的結果也是一個資料表（或資料行）。

在介紹篩選函數之前，需先釐清幾個概念，那就是關於表格式模型篩選的形式，可分為以下兩種：

- **內容篩選（Context Filter）**：表示與維度交叉時，為了計算特定資料格所套用的資料行與資料列條件（例如，圖8-1所點選的資料就被套入了「台灣」的篩選條件）。
- **明確篩選（Explicit Filter）**：是透過篩選控制項（交叉分析篩選器、樞紐分析表的篩選器、Power View的篩選面板…）所執行的資料篩選（例如，圖8-1整張樞紐分析表就被套入了「2013年」以及「Windows Phone」的篩選條件）。

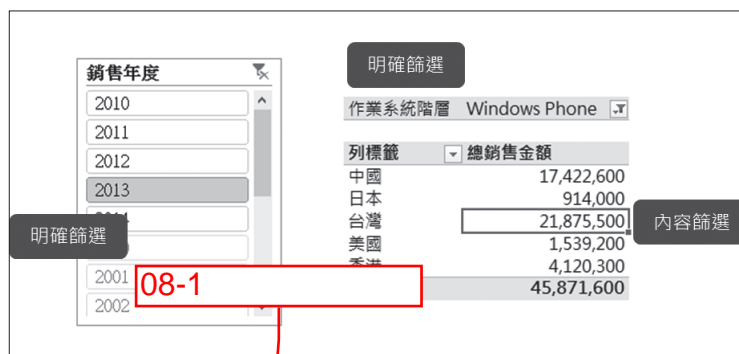


圖8-1：明確篩選與內容篩選

在不同的DAX篩選函數中，會針對不同的篩選形式產生不同的行為，這是在學習篩選函數時，必須特別注意的。

## Calculate 函數

在上一章與本章介紹DAX函數時，原則上是字母排序來依序介紹，以便讀者日後查詢。但在此則特別將Calculate函數拉到最前，因為它是最重要的篩選函數，稍後所說明的大部分範例幾乎都會出現Calculate函數的身影。

簡單來說，它就等於是T-SQL語法中的Select...Where子句的函數型態。當我們希望要計算在特定篩選條件下的彙總值，就可以使用Calculate函數。

函數介紹：根據指定的篩選條件式，評估值運算式的結果  
Calculate(<值運算式>,<篩選條件1>,<篩選條件2>...)

舉例來說，如果只想要看智慧型手機的銷售金額，就可以透過以下的Calculate函數來計算取得。它的用途就是限定計算範圍為「'產品大類別'[產品大類名稱]="智慧型手機"」，然後計算出「Sum('銷售明細'[銷售金額])」。所以在後面即將介紹的各項篩選函數，多半都能放在Calculate函數的篩選條件中。

DAX範例：智慧型手機銷售金額  
智慧型手機銷售金額:= Calculate(Sum('銷售明細'[銷售金額]),'產品大類別'[產品大類名稱]="智慧型手機")

## All函數

All函數是一種很特別的篩選函數，它的作用在於回傳指定資料表內的所有資料列，或指定資料行的所有值。也就是說，只要套用了All函數，就會清除所有的篩選條件（包括內容篩選與明確篩選）。All函數的概念與多維度分析中所有維度最上層的[All]成員的概念有點像，DAX中的All(‘資料表’)所代表的意義與MDX中[維度].[All]其實是非常相似的，因此，常被用於計算總計或佔全體比例時使用。

函數介紹：回傳資料表的所有資料列，或資料行的所有值

All( <資料表> )

All( <資料行1>[, <資料行2>[, <資料行3>[, …]] ] )

以下列範例來詳細說明All函數的使用模式。對於銷售金額這個欄位來說，最常見的量值寫法是「總銷售金額:=Sum('銷售明細'[銷售金額])」，這表示在進行動態分析時，這個量值會根據內容篩選，來計算個別維度條件下的銷售金額彙總（如圖8-2的第二個資料行）。當維度條件變化時，此量值的數字就會發生變動；但當我們希望計算在此維度條件下的佔總體百分比時，分母（總銷售金額）應該會是固定值，而非變動值。此時，就是使用All函數的最佳時機，只要使用了All函數，所計算的結果就不會受到維度條件的變動而產生影響。

DAX範例：佔全體銷售金額比例 (量值)

佔全體銷售金額比例:=Sum('銷售明細'[銷售金額])/ Calculate(Sum('銷售明細'[銷售金額]),All('銷售明細'))

在此，我們使用了Calculate函數，表示在評估全體資料的條件下（All('銷售明細')），去計算銷售金額的加總（Sum('銷售明細'[銷售金額])）來作為分母。



### 常發生的安裝錯誤與解決辦法

#### 總體百分比的不同寫法

關於總體百分比的計算，總共有2種主要寫法：

佔全體銷售金額比例:= Sum('銷售明細'[銷售金額])/ Calculate(Sum('銷售明細'[銷售金額]),All('銷售明細'))

佔全體銷售金額比例:= Sum('銷售明細'[銷售金額])/SumX(All('銷售明細'),'銷售明細'[銷售金額])



函數介紹：僅有指定的資料行套用篩選之外，回傳資料表的所有資料列，或資料行的所有值  
AllExcept (<資料表>,<資料行1>[,<資料行2>[,...]])

我們可以將AllExcept函數視為套用篩選的正面表列，並自行在指定的資料行範圍中設定篩選條件，以回傳出指定篩選條件下的全體資料。



DAX範例：

```
ALLEXCEPT銷售金額:=Calculate(Sum('銷售明細'[銷售金額]), AllExcept('客戶別','客戶別'[性別]))  
ALL銷售金額:= Calculate(Sum('銷售明細'[銷售金額]),All('銷售明細'))
```

08-4

以上述案例來說，我們在AllExcept ('客戶別','客戶別'[性別])中表示，只有性別將會套用篩選條件，而其他資料行則依舊移除篩選。對應到圖8-4，第一個表格顯示未做任何篩選條件下的總體資料。此時，你會發現「總銷售金額」與「ALLEXCEPT銷售金額」將得到相同的結果，這是因為兩者都套用了同樣「性別」的內容篩選。在第二個資料表中，排除了女性，「總銷售金額」與「ALL銷售金額」只顯示了男性的銷售金額，而「ALL銷售金額」則不受篩選影響。

08-4

至於圖8-4的第三個資料表，我們將篩選條件換成只保留大學以上的學歷，「ALL銷售金額」仍舊不受篩選影響，至於「ALLEXCEPT銷售金額」因為只有「性別」可例外套用篩選，因此，「學歷」的內容篩選對它將起不了任何作用。此時，「ALL銷售金額」與「ALLEXCEPT銷售金額」會得到相同結果。

列標籤	總銷售金額	ALL銷售金額	ALLEXCEPT銷售金額
女性	722,684,054	1,297,939,207	722,684,054
男性	575,255,153	1,297,939,207	575,255,153
總計	1,297,939,207	1,297,939,207	1,297,939,207

列標籤	總銷售金額	ALL銷售金額	ALLEXCEPT銷售金額
男性	575,255,153	1,297,939,207	575,255,153
總計	575,255,153	1,297,939,207	575,255,153

列標籤	總銷售金額	ALL銷售金額	ALLEXCEPT銷售金額
大學	310,424,717	1,297,939,207	1,297,939,207
五專	474,126,811	1,297,939,207	1,297,939,207
研究所	224,808,134	1,297,939,207	1,297,939,207
總計	1,297,939,207	1,297,939,207	1,297,939,207

08-4

圖8-4：AllExcept函數行為

## AllNoBlankRow函數

AllNoBlankRow函數是All函數的特殊形態，唯一的差別在於它的回傳結果不會包含空白資料列。AllNoBlankRow函數會回傳指定資料表內的所有非空白資料列，或指定資料行的所有非空白值，只要套用了AllNoBlankRow函數，也會清除所有的篩選條件（包括內容篩選與明確篩選）。

函數介紹：回傳資料表的所有非空白資料列，或資料行的所有非空白值

AllNoBlankRow(<資料表>)

AllNoBlankRow(<資料行>)

## AllSelected函數

之前我們在計算佔總體的百分比時，All函數可用來回傳總體資料表作為計算時的分子，且此回傳值是不會受任何篩選所影響的。但若我們想要計算的是佔2013年全年的總體百分比時，這表示我們的分母的「總體」觀念是被篩選過的，此時，又該如何計算呢？

這個時候就可以利用AllSelected函數，AllSelected函數只會清除所有的內容篩選，而套用明確篩選。若是不輸入引數（AllSelected()），則表示會套用「所有的」明確篩選；如果指定引數，則表示只套用與指定資料表或是資料行相關的明確篩選。

函數介紹：回傳移除內容篩選但套用明確篩選後資料表的所有資料列，或資料行的所有值

AllSelected()

AllSelected([<資料表>])

AllSelected([<資料行>])



### DAX範例：銷售金額比例(量值)

銷售金額比例:=Sum('銷售明細'[銷售金額])/Calculate(Sum('銷售明細'[銷售金額]),AllSelected())

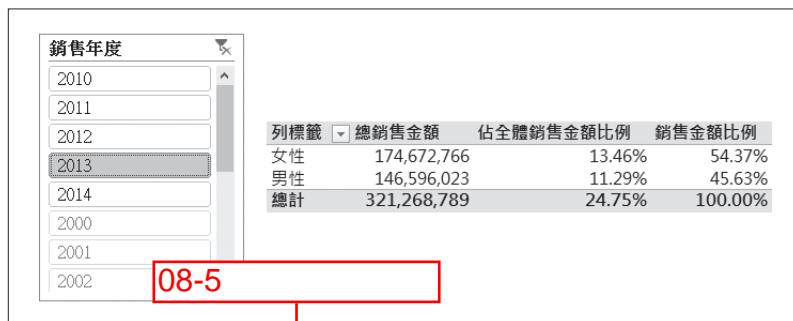


圖8-5：AllSelected函數行為

以圖8-5為例，我們透過交叉分析篩選器選取了2013年。由於All函數是不受任何篩選的影響，因此，女性的「佔全體銷售金額比例（分母是使用All函數）」為13.46%，其實表示的是2013年的女性銷售金額佔「有史以來」的銷售金額比例是13.46%。

至於女性的「銷售金額比例（分母是使用AllSelected函數）」為54.37%，則表示2013年的女性銷售金額佔「2013年」的銷售金額比例是54.37%。使用AllSelected函數的分母套用了明確篩選，也因此可以看到，「銷售金額比例」的合計值為100%，而非24.75%（2013年總銷售金額佔全體銷售金額的比例）。

### CalculateTable函數

CalculateTable函數與Calculate函數相似，唯一的差別是Calculate函數是回傳數值。至於CalculateTable是回傳資料表，它將指定資料表運算式根據條件篩選後的剩餘資料列，以表格的形式回傳。

函數介紹：根據指定的篩選條件式，評估資料表運算式的結果  
 CalculateTable (<資料表運算式>,<篩選條件1>,<篩選條件2>,...)

### Distinct函數

Distinct函數是根據指定的資料行內的值，進行去重複後，以單欄的資料表形式回傳。此函數與T-SQL中的SELECT DISTINCT子句的使用情境類似。

函數介紹：回傳去重複值清單  
 Distinct (<資料行>)



## Earlier函數

在一開始，我們曾強調DAX函數的結構雖然長得很像Excel公式，但兩者最大的差異在於DAX是以資料行為基礎的計算，而Excel公式則是以資料格為基礎的計算。由於資料行是一整批執行運算，而資料格則是逐筆進行計算，也因此在以資料行為基礎的計算時，很難去引用上一筆資料的內容（不像大家在Excel寫個公式，滑鼠一拉就能夠逐格套用公式）。

Earlier函數便是為了解決此問題所設計的DAX函數。它是以「逐列」的方式進行運算，且會從整個資料表中篩選出在目前資料列之前的資料列內容，來作為此次計算之基礎。所以要注意的是，Earlier函數可突破限制來達成許多複雜的遞迴式運算，但也正因它是逐筆執行，很容易造成效能上的問題，使用時必須謹慎。

Earlier函數通常用於計算資料行，通常是配合Filter函數等其他篩選函數一併使用，無法獨自存在。

函數介紹：將逐列執行所讀取的值暫存之結果

Earlier(<資料行>, <數值>)

此處的數值表示可以取得巢狀迴圈向外第n層讀取值的暫存結果，預設值是1。



DAX範例：

= CountRows (Filter('產品別', '產品別'[定價]>Earlier ('產品別'[定價])))+1

上報日	定價	作業系統平台	產品大類	定價排名	加入資料行
2010/9/1 ...	20800	iOS	智慧型手機	17	
2011/4/1 ...	23000	iOS	智慧型手機	8	
2012/6/1 ...	20400	Android	智慧型手機	19	
2012/12/2 ...	24500	iOS	智慧型手機	4	
2012/7/1 ...	21000	Windows Phone	智慧型手機	14	
2012/11/1 ...	16900	Windows Phone	智慧型手機	33	
2010/9/1 ...	23300	iOS	智慧型手機	7	
2011/4/1 ...	20400	iOS	智慧型手機	19	
2011/4/1 ...	9100	Android	智慧型手機	47	
2012/10/1 ...	25000	Other	平板電腦	2	
2012/5/1 ...	17500	Android	智慧型手機	29	
2012/10/2 ...	12000	Other	平板電腦	39	
2012/5/1 ...	22100	iOS	平板電腦	9	

圖8-6：使用Earlier函數計算定價排序值

08-6

## 08-6

我們以圖8-6的資料為例來說明。

**Step01：**Earlier函數會針對產品別資料表中的資料列進行逐筆計算，並取得對應'產品別'[定價]的值。舉例來說，第一筆資料回傳的定價值為19200元。

**Step02：**此時，透過Filter函數會回傳整個產品別資料表，評估這個資料表的每一筆記錄中的定價是否大於19200（先前逐筆計算時取回的暫存值）。

**Step03：**利用CountRows函數計算符合篩選資料的筆數，如果為0，表示這一筆記錄的定價是最高值。加上1，是為了讓第一名的排序值為1。以第一筆資料為例，計算出的排名為第17名。

**Step04：**進行下一列的計算，並重複上述步驟以產出對應的排序值。



### 常發生的部署錯誤與解決辦法

#### 排序值的不同寫法

關於排序值的計算，總共有2種主要寫法：

```
= CountRows(Filter('產品別', '產品別'[定價]>Earlier ('產品別'[定價]]))+1
```

```
=RankX(All('產品別'), '產品別'[定價])
```

兩者相較之下，當然是後者又容易維護而且效能佳。但因為Earlier函數解釋起來很複雜，因此，筆者才使用這個最簡單的應用來解釋這個函數的計算行為。當然，透過Earlier函數不只能進行排序，所有序列性或是遞迴性的運算都能夠彈性處理（例如，移動平均、累計平均、累加值…）。

## Earliest函數

Earliest函數與指定數值為1的Earlier函數，意義完全相同。

函數介紹：回傳

```
Earliest(<資料行>)
```

## Filter函數

Filter函數在觀念上與CalculateTable函數相似，都是回傳經指定條件篩選後的資料表結果。唯一的差別在於，CalculateTable函數可以支援多重篩選條件，而Filter只能支援一個，您可以將它視為簡化版的CalculateTable函數。

另一個關鍵的差異在於，Calculate及CalculateTable都能自動透過關聯性來評估篩選條件，即使篩選條件的資料行不位於指定資料表上也無所謂。但Filter函數中篩選條件所用的資料行，就必須存在於指定的資料表之上。

函數介紹：根據指定的篩選條件式，評估資料表運算式的結果  
Filter(<資料表>,<篩選條件>)

## Filters函數

Filters函數與Filter函數非常容易搞混，它們雖然長得很相像，但意義卻完全不同。前面介紹過的Filter函數比較像是「動詞」，它會根據篩選條件「執行」對指定資料表的篩選作業。至於現在要介紹的Filters函數比較像是「名詞」，它會回傳在指定資料行上被設定的篩選條件內容。簡單的說，如果我們將這個資料行設定篩選或是交叉分析篩選器，Filters函數就能如圖8-7般，自動偵測該資料行被選取的成員。

08-7

函數介紹：回傳指定資料行被套用篩選之資料  
Filters(<資料行>)



DAX範例：

選中的產品數量:=CountRows(Filters('產品別'[產品名稱]))



圖8-7 Filters函數

08-7 :

## HasOneFilter函數

在商業智慧分析中，有些資料行只允許單選而非複選，否則就會發生邏輯上的謬誤。舉例來說，像是匯率換算時要選定報表幣別，如果幣別複選，便無法產生正確的結果。為了判斷指定資料行套用的是否為單選的篩選條件，就需要利用HasOneFilter函數進行判斷。

函數介紹：若指定資料行所套用的直接篩選數為1時，則回傳True。

HasOneFilter(<資料行>)

HasOneFilter函數的意義與CountRows(Filters(<資料行>)) = 1的意義相同。

## HasOneValue函數

表格式模型的分析基礎是資料行，因此它無法像多維度模型般自動切換層級。舉例來說，多維度模型當顯示年的層級時，就顯示年的前期成長率；切換到月，則顯示月的前期成長率。對於表格式模型來說，年與月是不同資料行，所以是無法自動切換的。因此我們在分析時，需自行加入很多判斷條件，以避免出現不合理的情況。

舉例來說，當在計算前月成長率時，可能就需要判斷目前資料格對應至「月份」資料行是否僅對應到一個月份。如果現在顯示的是「年」的層級，資料格對應到「月份」資料行會有12個值，此時，比較月成長率就沒有意義了。所以此時可以利用HasOneValue函數，來判斷資料格對應至指定資料行時是否只有單一值。

函數介紹：若資料格對應至指定資料行的相異值數目為1時，則回傳True。

HasOneValue(<資料行>)

HasOneValue函數的意義與CountRows (Values(<資料行>)) = 1的意義相同。各位可能會覺得HasOneFilter函數與HasOneValue函數兩者相似度極高，其實兩者的差別在於HasOneFilter函數是用來判斷明確篩選，而HasOneValue函數則是用來判斷內容篩選。

## IsCrossFiltered函數

函數介紹：判斷指定資料行是否有套用交叉篩選。

IsCrossFiltered(<資料行>)

## IsFiltered函數

函數介紹：判斷指定資料行是否有套用直接篩選。

IsFiltered(<資料行>)

接下來，要介紹的兩個函數其實是非常接近的，都可用來判斷指定的資料行有無套用篩選。唯一的差別之處在於，IsCrossFiltered函數是判斷交叉篩選，而IsFiltered函數則是判斷直接篩選。

所謂的直接篩選指的是直接在指定的資料行上加入篩選條件，至於交叉篩選則是因為其他資料行（可以是在同個資料表或不同資料表）被設定了篩選條件後，造成指定的資料行內容發生變化。用文字描述感覺有點難理解，我們直接

以範例來說明。



DAX範例：

IsFilter產品名稱:= IsFiltered('產品別'[產品名稱])

IsCrossFilter產品名稱: 08-7 名稱])

首先，筆者在產品別資料表利用以上公式加入了兩個量值，所指定的都是「產品名稱」資料行，各位可以比對圖8-7的結果。圖中是將這兩個量值與「作業系統平台」資料行做交叉，並設定「產品名稱」作為交叉分析篩選器。

圖8-8上方表示完全沒有篩選「產品名稱」（全選），由於IsFiltered函數是判斷直接篩選，因此，它全部顯示為False，這是因為「產品名稱」沒有被設定任何篩選條件。但此時，IsCrossFiltered函數與各個作業系統平台交錯的資料格卻顯示True，總計卻顯示False，這是因為IsCrossFiltered函數是判斷交叉篩選，因此，該資料格是與作業系統平台做交叉，等於是套用了內容篩選，才會顯示True。至於總計的部分就沒有內容篩選的問題，因此，就會顯示False。

產品名稱

Acer Allegro  
Acer ICONIA Tab ...  
Apple iPad 2 3G 1...  
Apple iPad 2 3G 3...  
Apple iPhone 4 16...  
Apple iPhone 4 16...  
Apple iPhone 4 32...  
Apple iPhone 4 32...

列標籤

IsCrossFilter產品名稱	IsFilter產品名稱
Android	TRUE
iOS	TRUE
Other	TRUE
Windows Phone	TRUE
總計	FALSE

產品名稱

ASUS Eee Pad Tra...  
ASUS PadFone  
ASUS TAICHI  
ASUS Transformer...  
HTC 7 Mozart  
HTC Desire C

列標籤

IsCrossFilter產品名稱	IsFilter產品名稱
Android	TRUE
iOS	TRUE
Other	TRUE
Windows Phone	TRUE
總計	TRUE

08-8

08-8

圖8-8：IsFiltered函數與IsCrossFiltered函數的行為差異

再來看圖8-8下方，我們取消選取前3款手機，所以「產品名稱」被套用直接篩選了，也因此無論IsFiltered函數或是IsCrossFiltered函數都會顯示True。

## Related函數

Related函數通常用於具有「外部索引鍵」的資料表，並可根據鍵值的對應關係，回傳「唯一」的指定資料行（通常是位於主索引鍵的資料表上）的值。在本書《第5-4-4節：雪花狀維度設計》中曾介紹如何利用Related函數來解決雪花狀維度的設計議題，另外在《第6-2-1節：多對多維度關係設計》中，也說明了如何使用Related函數，來處理多對多維度議題。

函數介紹：回傳根據鍵值關係所對應的資料行唯一值  
Related(<資料行>)

不粗

08-9

位資料	上開日	EndDate	售價	售價排名	作業系統平台	產品大類
位資料	2010/9/1 ...		20800	3	iOS	智慧型手機
位資料	2011/4/1 ...		23000	12	iOS	智慧型手機
位資料	2012/6/1 ...		20400	11	Android	智慧型手機
位資料	2012/7/1 ...		21000	14	Windows Phone	智慧型手機
位資料	2012/11/1 ...		16900	37	Windows Phone	智慧型手機
位資料	2010/9/1 ...		23300	2	iOS	智慧型手機

圖8-9：使用Related函數設計雪花狀維度

## RelatedTable函數

RelatedTable函數的使用方法剛好與Related函數相反，Related函數通常用於具有「外部索引鍵」的資料表，以回傳具主索引鍵資料表的唯一值。而RelatedTable函數則常用於具有「主索引鍵」的資料表，以傳回對應的多筆記錄。

函數介紹：回傳根據鍵值關係所對應的資料表

RelatedTable(<資料表>)

## UseRelationship函數

在表格式模型設計的過程中，我們可以透過建立「關聯性」的方式來建立資料表之間的關聯，而資料模型就會根據此關聯性，來計算跨資料表的計算行為。但當要查詢的資料表間尚未設定關聯性，此時，便可透過UseRelationship函數來建立臨時性的關聯性。所以此函數並沒有回傳值，而是用來標示資料表間的關聯性。

要注意的是，UseRelationship函數不能獨立使用，僅可用於將篩選作為引數的函數，例如：Calculate、CalculateTable、ClosingBalanceMonth、ClosingBalanceQuarter、ClosingBalanceYear、OpeningBalanceMonth、OpeningBalanceQuarter、OpeningBalanceYear、TotalMTD、TotalQTD、TotalYTD等函數。

UseRelationship函數支援巢狀架構，但在每個計算資料行或量值中，最多只能使用10次UseRelationship函數。

函數介紹：標示透過指定資料行所建立的資料表間關聯性

UseRelationship(<外部索引鍵資料行>,<主索引鍵資料行>)

01

02

03

04

05

06

07

08-01

進階DAX函數



DAX範例：計算總銷售金額(假設銷售明細資料表與銷售日別資料表間尚未建立關聯性)

```
總銷售金額:=Calculate(Sum('銷售明細'[銷售金額]), UseRelationship('銷售明細'[銷售日], '銷售日別'[資料日]))
```

## Values函數

Values函數與Distinct函數很類似，都是用來回傳指定資料行的去重複值結果，唯一的差異在於，Distinct函數的回傳結果不會包括空白值，而Values函數會包含因為鍵值無法正確對應時，所產生的空白值（類似多維度模型中的未知成員）。

函數介紹：回傳指定資料行的去重複值列表  
VALUES(<資料行>)

## 8-1-2 路徑函數

不粗

到目前為止，各位應該對於路徑函數應該已不陌生，因為在《第6-1-1節：父子式階層》以及《第6-1-2節：會計科目階層》中，我們已經花了很多篇幅來介紹路徑函數的應用。需注意的是，所有的路徑函數都無法在DirectQuery模式中使用。

## Path函數

Path函數是路徑函數應用的基礎，它會根據指定的鍵值與父成員的資料行，自動組成遞迴字串，以表示其父子式階層的對應關係。

函數介紹：回傳  
Path<鍵值資料行名稱>, <父成員資料行名稱>)



DAX範例：根據主管編號產生路徑字串

```
=Path([EmployeeId],[ManagerId])
```



08-10

EmployeeId	ManagerId	組織架構路徑
1	1	1
101	106	1 106 101
106	1	1 106
110	101	1 106 101 110
111	101	1 106 101 111
112	101	1 106 101 112

圖8-10：產生路徑字串

## PathContains函數

PathContains函數是用來判斷指定的物件（索引鍵值，可以是數值或字串）是否存在於指定的路徑字串中。

函數介紹：判斷指定路徑物件是否存在於路徑字串中

PathContains(<路徑字串>, <物件>)

## PathItem函數

PathItem函數可協助我們根據路徑字串，回傳指定位置的物件識別碼，因此多用於設計父子式階層的過程中，來判別指定層級的成員內容。

函數介紹：回傳

PathItem(<路徑字串>, <位置>[, <型態>])

可用值	替代可用值	敘述
TEXT	0	將物件識別碼以字串形式回傳，此為預設值
INTEGER	1	將物件識別碼以整數形式回傳



DAX範例：回傳路徑字串第一層成員的名稱

=LOOKUPVALUE([員工名稱],[EmployeeId],PATHITEM([組織架構路徑],1,INTEGER))

01

02

03

04

05

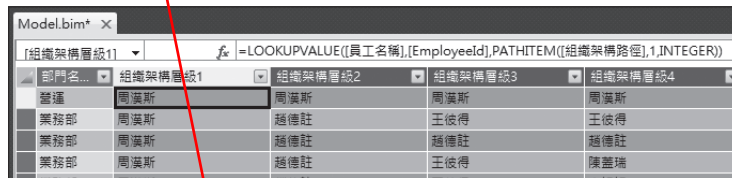
06

07

08-01

進階DAX函數

08-11



部門名	組織架構層級1	組織架構層級2	組織架構層級3	組織架構層級4
營運	周漢斯	周漢斯	周漢斯	周漢斯
業務部	周漢斯	趙德註	王彼得	王彼得
業務部	周漢斯	趙德註	趙德註	趙德註
業務部	周漢斯	趙德註	王彼得	陳蓋瑞

圖8-11：回傳路徑字串各層成員的名稱

## PathItemReverse函數

PathItemReverse函數的作用與PathItem函數相似，唯一的差別之處在於PathItem函數的層級數是從頂層（Root）向下計算，至於PathItemReverse函數，則是逆向由葉層（Leaf）向上計算。

函數介紹：回傳

PathItemReverse(<路徑字串>, <位置>[, <型態>])

## PathLength函數

由於路徑函數是以「|」作為分隔符號來連結各路徑物件。透過PathLength函數可協助我們判定該路徑字串中所包含的物件數，並計算該指定路徑之深度（可參考《圖6-11：計算「目前層級」與「最深層級」》）。

函數介紹：回傳路徑包含之成員數

PathLength(<路徑字串>)

不粗

## 08-02 DAX陳述句查詢

在之前的介紹中，我們都把焦點放在利用DAX來產生計算資料行以及量值。在多維度模型中的MDX語言，可用來產生導出成員與量值，同時也提供MDX陳述句來查詢Cube。表格式模型既然身為全新型態的商業智慧資料源，它當然也需要一套完整的查詢語法來協助存取資料模型。因此，在DAX中也提供了DAX陳述句，可用來執行資料查詢。

### 8-2-1 查詢工具

目前支援DAX查詢的工具相當有限，以官方工具來說，如下所列：

- Power View
- SQL Server Management Studio
- Reporting Services查詢設計工具

其中，Power View是完全基於DAX查詢的視化工具，目前Power View僅支援表格式模型與PowerPivot。雖然微軟的終極目標是希望能同時讓多維度模型與表格式模型兩種兼容MDX以及DAX。但以目前的架構來看，僅有表格式模型可以同時兼容兩種語法，多維度模型現在仍無法使用DAX查詢（目前微軟已推出支援DAX的多維度模型技術預覽版，但目前尚未得知明確的推出時間）。

但因Power View並不提供讓使用者自行輸入DAX查詢的機制，所以後續的範例將是以SQL Server Management Studio為查詢工具。只需在SSMS新增「Analysis Services MDX查詢」，即可連結表格式模型並開始撰寫DAX查詢語法。

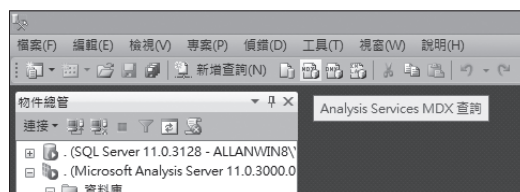


圖8-12：新增DAX查詢

08-12

08-13

要注意的是，雖然可以透過SSMS執行DAX查詢，但目前SSMS的物件模型、偵錯與Intellisense都還是基於MDX，不但畫面上的物件清單還是以多維度的概念呈現，就算我們寫的是正確的DAX語法，仍舊會在SSMS上顯示錯誤，如圖8-13。



圖8-13：執行DAX查詢

另一個常見的DAX應用來自於Reporting Services。因為DAX的最大特色在於可同時兼顧明細與彙總查詢，再加上它是透過記憶體運算，效能會比直接查詢原始資料要來得快速許多。目前SQL Server 2012的SSDT正式廢除了報表模型專案（舊的報表模型仍舊相容，只是無法新增報表模型專案），看來微軟希望引導報表模型的使用者改用表格式模型的態勢，是越來越明朗的。

關於在Reporting Services上使用DAX這點，不得不對微軟有點微詞，既然表格式模型是SQL Server 2012推動的技術主力，但許多配套措施卻做得不夠完善，像是剛才提到的SSMS無法進行DAX偵錯，在Reporting Services的查詢設計工具中（需資料來源為連至Analysis Services）若是輸入DAX語法，竟然會彈出「請確認查詢為MDX而非DMX」的錯誤訊息。

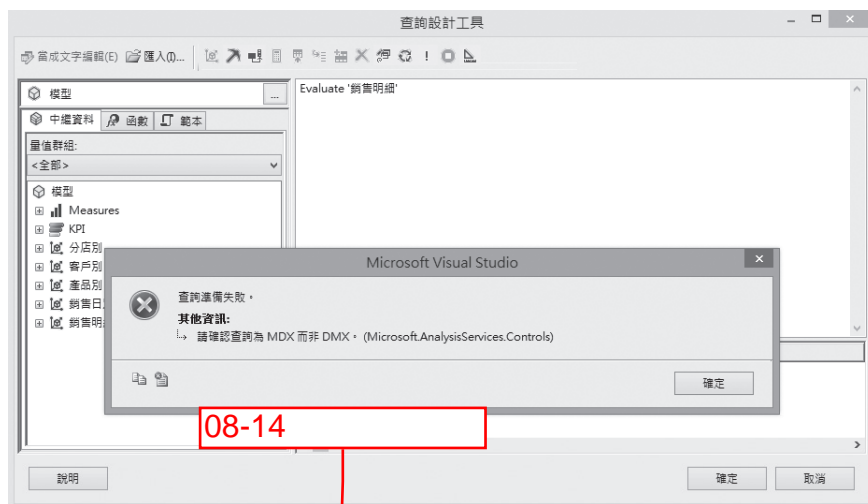


圖8-14：查詢設計工具錯誤訊息

你沒看錯！是「DMX」而非「DAX」。「DMX」是Analysis Services中的資料採礦查詢語言。會發生這個錯誤訊息是因為Reporting Services查詢設計工具中，有針對MDX語法做結構上的驗證，如果驗證不過，就視為DMX。因為微軟在SQL Server 2012中，並未修改此部分的驗證，所以才會跑出這樣的錯誤訊息。既然如此，筆者就試著將查詢設計工具切換為DMX模式（點選工具列上鶴嘴鉤圖示），並點選「設計模式」，便可以自行輸入DAX語法。

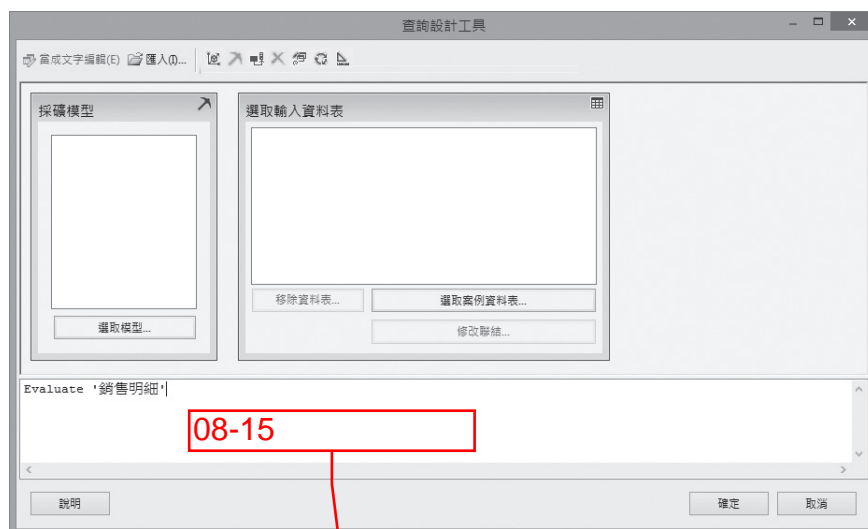


圖8-15：在DMX模式下輸入DAX

點選「確定」後，切換至「欄位」分頁，果然Reporting Services成功執行了DAX查詢並取出其欄位資訊。所以在微軟還沒解決這個Bug之前，各位讀者還需要忍受一段「DMX？DAX？傻傻分不清楚！」的日子。

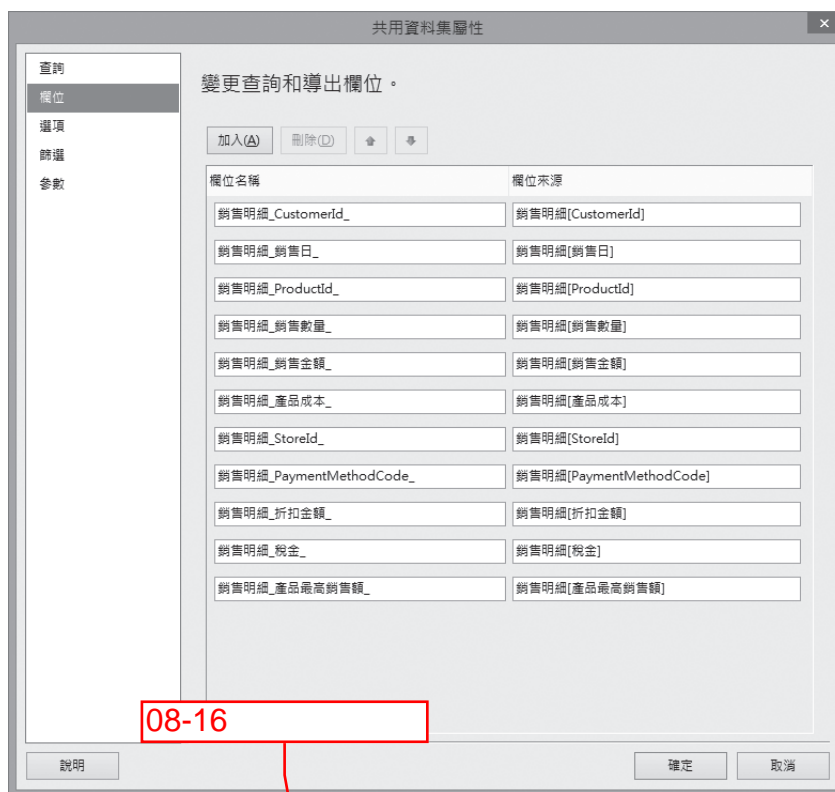


圖8-16：成功讀取DAX的資料行資訊

## 8-2-2 DAX陳述句基本句型

DAX陳述句的句型範本如下：

函數介紹：DAX陳述句基本句型

```
[DEFINE { MEASURE <資料表名稱>[<量值名稱>] = <運算式> }
EVALUATE <資料表名稱或資料表運算式>
[ORDER BY {<運算式> [{ASC | DESC}]}[, ...]
[START AT {<值>|<參數>} [, ...]]]
```

從句型範本來看，可將DAX陳述句分成4個區段：

- **Define**區段：用來自訂量值計算。
- **Evaluate**區段：這是DAX查詢最重要的核心，在此用來定義回傳查詢結果的資料表結構。
- **Oder By**區段：用來定義資料的排序模式。
- **Start At**區段：用來定義DAX查詢之參數。

最單純的DAX查詢僅需透過Evaluate區段即可達成。首先，來看最簡單的DAX範例：

函數介紹：查詢銷售明細資料表  
Evaluate '銷售明細'

銷售明細[CustomerID]	銷售明細[SalesTime]	銷售明細[ProductID]	銷售明細[SalesAmount]	銷售明細[ProductSalesAmount]	銷售明細[ProductSalesAmount]	銷售明細[StoreID]
101008544	2012/12/28 上午...	(2C1C3D7B-07...	1	25000	19000	103
107005103	2014/7/9 上午 1...	(C89D799F-DE...	1	5100	2700	107
104006482	2013/2/28 上午...	(D6779118-C43...	1	22100	15800	103
208002168	2011/6/10 上午...	(9A43AC6A-52...	1	6900	3600	206
504008400	2012/6/30 上午...	(RCCC80BE-7...	1	20400	14800	504
		02-FA...	1	6000	2500	602
		3F-C1...	1	18820	10200	204
301004494	2014/8/8 上午 1...	(B63DAAE8-B...	1	13700	8900	301
703005712	2013/3/12 上午...	(E51DDF47-C9...	1	13300	6600	703

08-17

圖8-17：查詢「銷售明細」資料表

在此與DAX公式相同，都是透過單引號的方式來宣告資料表。若您是使用左側的物件清單拖拉，由於物件清單仍是MDX，拉出來的資料表名稱會是帶中括號([銷售明細])，必須手動修正。

如果我們只需要看到2013年的銷售明細，就可以根據這個基本型態再套上篩選，此時，陳述句會變為：

函數介紹：查詢2013年度銷售明細  
Evaluate  
CalculateTable('銷售明細','銷售日別'[年度]=2013)

```
Evaluate
Filter('銷售明細',Related('銷售日別'[年度])=2013)
```

其中，**CalculateTable**可以指定其他資料表上的資料行來作為篩選條件，至於**Filter**函數中篩選條件所用的資料行，就必須存在於指定的資料表之上。為了解決這個限制，筆者使用了**Related**函數取回銷售年度資料，如此一來，就能夠執行篩選條件判斷了。

若我們希望回傳的結果能夠先根據銷售日期排序，然後再根據金額由高至低排序的話，那麼就需要使用**Order by**區段的句型。原則上，**程式框**都只能夠處理同一資料表上資料行的排序邏輯，由於銷售日期是存放在「銷售日別」資料表上，因此，筆者也是透過**Related**函數，來解決這個問題。

```
函數介紹：查詢2013年度銷售明細，並依照銷售日期(升冪)與銷售金額(降冪)排序
Evaluate
CalculateTable('銷售明細','銷售日別'[年度]=2013)
Order by Related('銷售日別'[資料日]), '銷售明細'[銷售金額] Desc
```

除了明細分析之外，**DAX**查詢的一大賣點就是彙總分析。相較於傳統**T-SQL**必須處理跨資料表間的**Join**造成語法解構複雜，由於存取的標的是記憶體中的儲存體，因而可獲得大幅度的效能提升。在**DAX**中，要處理資料表彙總分析，就需使用到**Summarize**函數。我們先複習一下**Summarize**函數的結構。

```
函數介紹：回傳彙總後的摘要資料表
Summarize(<資料表>, <groupBy資料行>[, <groupBy資料行>]...[, <資料行名稱>, <值運算式>]...)
```

當希望查詢各產品大類各年度的銷售金額與銷售數量的話，就可以利用**Summarize**函數來產生彙總結果，如圖8-18。

```
函數介紹：查詢各產品大類各年度的銷售金額與銷售數量
Evaluate
Summarize (
    '銷售明細',
    , '銷售日別'[年度]
    , '產品別'[產品大類]
    , "總銷售金額", Sum('銷售明細'[銷售金額])
    , "總銷售數量", Sum('銷售明細'[銷售數量])
)
Order by '銷售日別'[年度], '產品別'[產品大類]
```



MDXQuery3.mdx - ...ALLANWIN8\Yiin\* X

Cube: 模型

量值群組: 模型

Measure

KPI

分店別

客戶別

產品別

銷售日別

銷售明細

折扣金額

產品成本

產品最高銷售額

税金

銷售金額

銷售數量

Evaluate

```
Summarize (
    '銷售明細'
    , '銷售日別'[年度]
    , '產品別'[產品大類]
    , "總銷售金額", SUM('銷售明細'[銷售金額])
    , "總銷售數量", SUM('銷售明細'[銷售數量])
)
Order by '銷售日別'[年度], '產品別'[產品大類]
```

100 %

銷售日別[年度]	產品別[產品大類]	[總銷售金額]	[總銷售數量]
2010	平板電腦	1083600	63
2010	智慧型手機	17453570	947
2011	平板電腦	52947800	3395
2011	智慧型手機	113542729	8019
2012	平板電腦	116647600	6947
2012	智慧型手機	371064946	24309
2013	平板電腦	79193930	5080
2013	智慧型手機	242074859	20176
2014	平板電腦	71924985	5541
2014	智慧型手機	232005188	19600

08-18

圖8-18：查詢彙總資訊

除了基礎的彙總函數，我們還可以透過之前所學過的DAX函數再進行擴充。例如：

- 利用DistinctCount函數計算去重複客戶數。
- 利用AllExcept函數計算各年度的總銷售量，並以此數值作為分母來計算銷售比例。

以上情境的語法範例如下：

函數介紹：查詢各產品大類各年度的銷售金額、銷售比例與去重複客戶數

```
Evaluate
Summarize (
    '銷售明細'
    , '銷售日別'[年度]
    , '產品別'[產品大類]
    , "總銷售金額", Sum('銷售明細'[銷售金額])
    , "年度總銷售金額", Calculate(Sum('銷售明細'[銷售金額]), AllExcept('銷售明細', '銷售日別'[年度]))
    , "銷售金額%", Sum('銷售明細'[銷售金額])/Calculate(Sum('銷售明細'[銷售金額]), AllExcept('銷售明細', '銷售日別'[年度]))
    , "去重複客戶數", DistinctCount('客戶別'[CustomerId])
)
Order by '銷售日別'[年度], '產品別'[產品大類]
```

08-19

執行結果如圖8-19。如果是利用查詢，除了必須要處理銷售明細、銷售日別、產品別、客戶別等四個資料表間的Join，同時還必須先計算出各年度的總銷售金額，才有可能算出年度銷售比例，更別提要計算去重複客戶數所耗用的計算資源。但透過DAX查詢，只需使用Summarize函數就能輕鬆搞定。各位還可以看到圖8-19右下角顯示的執行速度，這正是記憶體運算的強大之處。

08-19

08-19

銷售日別[年度]	產品別[產品大類]	銷售金額	年度總銷售金額	銷售金額%	去重複客戶數
2010	平板電腦	1003000	18537170	0.054155247645676	63
2010	智慧型手機	17453570	18537170	0.941544752354322	682
2011	平板電腦	52947800	166490529	0.318022894963887	2679
2011	智慧型手機	113542729	166490529	0.681977105136113	4232
2012	平板電腦	116647800	487712546	0.239172830804621	4519
2012	智慧型手機	371064946	487712546	0.760827169185579	7335
2013	平板電腦	79193930	321268789	0.246503655240447	3897
				0.753496344759553	7014
				0.23664970243017	3572
				0.76335029756983	6217

圖8-19：計算年度銷售比例

有時在商業分析中，我們並不需要取回全部的結果，而是希望根據條件找出前幾筆或是後幾筆的資料，例如「哪些是毛利率最高的前10種商品？」、「哪些是業績達成率最差的分店？」，此時，就可以搭配使用TopN函數。

函數介紹：查詢2013年毛利率最高的前10款手機

```

Evaluate
TopN(10,
Summarize (
    CalculateTable('銷售明細', '銷售日別' [年度]=2013)
    , '產品別' [產品名稱]
    , "毛利率", [毛利率]
)
, [毛利率])
Order by [毛利率] Desc
  
```

函數介紹：查詢2013年毛利率最低的10款手機

```

Evaluate
TopN(10,
Summarize (
    CalculateTable('銷售明細', '銷售日別' [年度]=2013)
  
```

```

, '產品別'[產品名稱]
, "毛利率", [毛利率]
)
, [毛利率], 1)
Order by [毛利率]

```

MDXQuery3.mdx - ...ALLAN\WIN8\Yiin\* x

Cube: 模型

量值群組: <全部>

模型

- Measures
- KPI
- 分店別
- 客戶別
- 產品別
- 銷售日別
- 銷售明細

100 %

訊息 結果

產品別[產品名稱]	[毛利率]
Apple iPhone 5 8GB	0.623618438891205
MOTO MOTOSMART ...	0.578709855041237
LG Optimus L7	0.528165041502136
HTC 7 Mozart	0.526488899330436
NOKIA Lumia 610	0.510565593087196
HTC Rhyme 音韻機	0.504784097087009
Apple iPhone 4 16GB	0.504426336727553
Apple iPhone C	0.5
Sony Xperia sola	0.497801317460373
NOKIA Lumia 800	0.4953785768549

08-20

圖8-20：查詢2013年毛利率最高的前10款手機

### 8-2-3 DAX陳述句進階句型

在上一節中，我們介紹了DAX的陳述句基礎語法。基本上，它是基於既有的表格式模型架構進行查詢，但DAX的能力遠不只於此，透過DAX陳述句可以讓分析人員任意修改模型結構，包括加入計算資料行或量值。

若要在DAX陳述句中加入自訂量值，就必須使用Define Measure句型。這與多維度模型MDX語法中的Create Measure句型類似，都可以產生自訂量值，然後讓後續的語法參照此自訂量值，以擴充計算。在以下範例中，我們將產生「去重複客戶數」自訂量值，然後再利用銷售金額與去重複客戶數計算出「人均年度消費」的結果，DAX語法範例如下：

函數介紹：自訂「去重複客戶數」量值

```
Define Measure '銷售明細'[去重複客戶數] = DistinctCount('客戶別'[CustomerId])
```

Evaluate

```
Summarize (
    '銷售明細'
    , '銷售日別'[年度]
    , '產品別'[產品大類]
    , "總銷售金額", Sum('銷售明細'[銷售金額])
    , "去重複客戶數", [去重複客戶數]
    , "人均年度消費", [總銷售金額]/[去重複客戶數]
)
```

```
Order by '銷售日別'[年度], '產品別'[產品大類]
```

MDXQuery3.mdx - ...ALLANWIN8\Yiin)\* X

Cube: 模型

量值群組: 全部

模型

- Measures
- KPI
- 分店別
- 客戶別
- 產品別
- 銷售日別
- 銷售明細

```
Define Measure '銷售明細'[去重複客戶數] = DistinctCount('客戶別'[CustomerId])

Evaluate
Summarize (
    '銷售明細'
    , '銷售日別'[年度]
    , '產品別'[產品大類]
    , "總銷售金額", Sum('銷售明細'[銷售金額])
    , "去重複客戶數", [去重複客戶數]
    , "人均年度消費", [總銷售金額]/[去重複客戶數]
)
Order by '銷售日別'[年度], '產品別'[產品大類]
```

銷售日別[年度]	產品別[產品大類]	[總銷售金額]	[去重複客戶數]	[人均年度消費]
2010	平板電腦	1083600	63	17200
2010	智慧型手機	17453570	682	25591.7448680...
2011	平板電腦	52947800	2679	19764.0164240...
2011	智慧型手機	113542729	4232	26829.5673440...
2012	平板電腦	116647600	4519	25812.7019252...
2012	智慧型手機	371064946	7335	50588.2680299...
2013	平板電腦	79193930	3697	21421.133351366
2013	智慧型手機	246924250	7014	34513.0965212...
2014	平板電腦	252005188	3572	20135.7740761...
2014	智慧型手機	252005188	6217	37317.8684252...

圖8-21：自訂量值

不只是自動量值，透過DAX甚至可在資料表中，加入自訂計算資料行。此時就要透過在上一章中介紹過的AddColumns函數。在以下的範例中，我們將利用AddColumns函數在「銷售明細」資料表中產生「星期幾」計算資料行，然後在Summarize函數的彙總計算中，引用這個自訂計算資料行來作為彙總資料行，以計算出一周內，每日的銷售金額彙總值。

需

函數介紹：自訂「星期幾」資料行

Evaluate

Summarize (

AddColumns('銷售明細',"星期幾",WeekDay(Related('銷售日別'[資料日])))

,[星期幾]

, '產品別'[產品大類]

, "總銷售金額", Sum('銷售明細'[銷售金額]))

)

Order by '產品別'[產品大類],[星期幾]

需要注意的是，在後續引用自訂計算資料行時（無論是Summarize函數或是排序），都無須指定資料表名稱。因為只有模型中實體存在的資料行，才能夠在前方標記資料表名稱。

MDXQuery3.mdx - ...ALLANWIN8(Yiin)\* X

Cube:

模型

中繼資料 函數

量值群組

全部

模型

Measures

KPI

分店別

客戶別

產品別

銷售日別

銷售明細

```
Evaluate
Summarize (
  AddColumns('銷售明細',"星期幾",WeekDay(Related('銷售日別'[資料日])))
  , [星期幾]
  , '產品別'[產品大類]
  , "總銷售金額", Sum('銷售明細'[銷售金額]))
Order by '產品別'[產品大類],[星期幾]
```

100 %

訊息 結果

[星期幾]	產品別[產品大類]	[總銷售金額]
1	平板電腦	47491580
2	平板電腦	46899940
3	平板電腦	45684370
4	平板電腦	45795285
5	平板電腦	44206120
6	平板電腦	46211865
7	平板電腦	45508755
1	智慧型手機	143512438
2	智慧型手機	143151479
3	智慧型手機	133225701
4	智慧型手機	140046242
5	智慧型手機	97467
6	智慧型手機	27691
7	智慧型手機	139980274

08-22

圖8-22：自訂計算資料行

將自訂模型結構做到最極致，那就是在DAX語法中可以利用Row函數來產生自訂資料列，透過此函數可自行指定各個資料行的名稱及值運算式。在以下的範例中，就是透過Row函數來產生記錄2013年銷售狀況的自訂資料列。

01

02

03

04

05

06

07

08-02

DAX陳述句查詢

函數介紹：自訂資料列

Evaluate

```
Row(  
  "名稱", "2013年總計銷售",  
  "數值", Calculate(Sum('銷售明細'[銷售金額]), '銷售日別'[年度]=2013)  
)
```

自訂資料列還可以進一步透過CrossJoin函數，擴充成為自訂資料表。

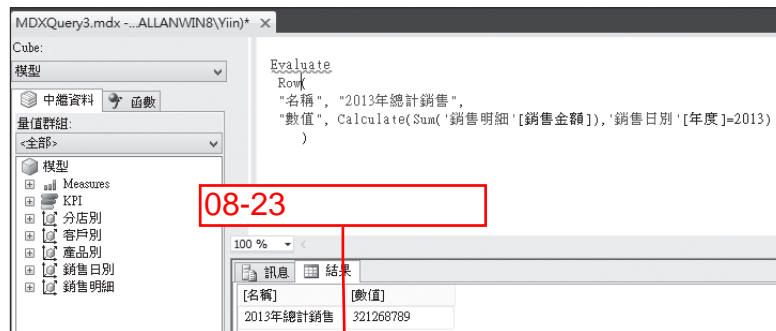


圖8-23：自訂資料列

看來微軟已經為DAX這個新語言設計了非常完整的架構，讓人非常期待基於DAX的Power View以及利用表格式模型取代報表模型後的Reporting Services，微軟會基於DAX技術提供什麼樣的新的驚喜，就讓我們拭目以待吧！◆◆