SQL Server 2012
商業智慧2.0

第06章進階BISM 表格式模型設計

進階維度設計
進階商業分析設計
檢視方塊與資料分割

在前一章中，介紹了基礎的表格式模型設計概念，但當我們需要一些更複雜的商業需求時，表格式模型是否能夠展現出更強大的設計彈性呢？接著將在本章繼續介紹進階的模型設計技巧。

06-01 進階維度設計

在上一章中，我們介紹了基本的維度與階層設計，包括了一般維度與時間維度，在本小節中，我們將更深入的介紹以往多維度模型中有包含的特殊維度結構的處理模式。

6-1-1 父子式階層

傳統的維度階層概念中，若這個階層包含3個層級，那麼就需要利用3個資料行來建構，如此一來，當層級數過多，就會造成資料準備上的困擾。這種狀況普遍的出現在人事架構、會計科目以及製造業的物料需求表（BOM）。為了簡化設計模式，在多維度模型中提出了**父子式階層**（Parent-Child Hierarchy）的概念來解決這種問題（層級數過多、層級數未知、層級數經常異動…）。

父子式階層的最重要精神在於只需維護兩個資料行：「成員鍵值」與「父成員鍵值」，就能重建出上下對應的階層關係。而不必事先考量階層的層級數量。可說是一種非常彈性的階層設計模式。

在表格式模型中，**目前並未「直接」提供父子式維度的實作方法**，但是在DAX語法中，則提供了一系列的「**路徑函數**」，可以做到「**近似**」父子式階層的效果，所謂的「近似」表示還有諸多限制是目前表格式模型所做不到的。列舉如下：

- 表格式模型必須透過路徑函數作為中介，才能產生父子式階層結構，而多維度模型不需要。
- 表格式模型的父子式階層數量必須透過事前設定，但多維度模型可以動態生成（也就是說層級數極多的人事階層維度目前並不建議使用表格式模型）。
- 表格式模型的父子式階層目前不直接支援非葉層資料隱藏的效果，多維度模型則可自由切換。

則

- 表格式模型的父子式階層目前不直接支援一元運算子計算功能（在本章中，我們會透過DAX來做到近似效果），多維度模型則支援。
- 表格式模型的父子式階層為提供會計科目範本（在本章中，我們會透過DAX來做到近似效果），但多維度模型中提供。

關於上述提到的「非葉層資料」、「一元運算子」…等概念，我們將會於後續的內容中詳細說明。不過，還是要提醒各位，表格式模型本身的定位是用來簡化多維度模型的開發，而非用來取代多維度模型，也因此，在這個版本中的父子式維度的確有許多不足之處，這是各位在選擇開發模式之前，必須要事前考量清楚的。

在本書範例中（Ch06/組織架構）的資料模型中加入了「組織架構」資料表。其中的主索引鍵為「EmployeeId（員工編號）」，其中同時提供了「ManagerId（主管員工編號）」以建構出父子階層的架構。接下來，示範如何利用這個組織架構資料表，來產生組織架構的父子式階層。

應該是06-1
我這章的圖號都弄錯了
請幫我改正

圖6-01：組織架構資料表

首先，表格式模型必須透過DAX路徑函數建構出完整的組織架構路徑，其DAX函數的範例如下：

```
DAX範例： 建構父子式階層路徑

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

Model.bim* ×

[組織架構路徑] =Path([EmployeeId],[ManagerId])

EmployeeId	ManagerId	E...	員工...	職稱	雇用日	性...	部門名...	組織架構路徑
1		A001	周漢斯	CEO	1999/2/15	M	營運	1
101	106	A101	王彼得	業務部協理	2001/2/4	M	業務部	1|106|101
106	1	A106	趙德註	業務副總	1999/6/4	M	業務部	1|106
110	101	A110	陳蓋瑞	業務經理	2001/7/1	M	業務部	1|106|101|110
111	101	A111	李娟娟	業務經理	2001/7/1	F	業務部	1|106|101|111
112	101	A112	林志鈴	業務經理	2001/7/1	F	業務部	1|106|101|112
113	101	A113	鍾湯尼	業務經理	2001/7/1	M	業務部	1|106|101|113
114	101	A114	馬蓋先	業務經理	2001/7/1	M	業務部	1|106|101|114
115	101	A115	吳愛愛	業務經理	2001/7/1	F	業務部	1|106|101|115
116	106	A116	比爾蓋	業務部協理	2001/7/1	M	業務部	1|106|116
117	116	A117	謝阿志	業務經理	2001/7/1	M	業務部	1|106|116|117
118	116	A118	李大偉	業務經理	2001/7/1	M	業務部	1|106|116|118

圖6-02：建構父子式階層路徑

06-2

不粗

各位可以發現利用函數，就可以建構出從根節點到指定節點間的完整路徑，並以「|」符號相連接，如圖6-02。

06-2

在建構出完整路徑後，便能使用PATHITEM函數來取出指定層數的節點編號（從1開始計數）。**若是對於PATHITEM的使用方式有任何疑問，請參閱本書《第7-1-2節：路徑函數》**。

```
DAX範例：根據父子式階層路徑回傳指定層級的成員編號

= PATHITEM([組織架構路徑],1,INTEGER)
```

而回傳的節點編號搭配LOOKUPVALUE函數，就可以將員工編號轉換為對應的員工名稱了，如圖6-03。

```
DAX範例：根據父子式階層路徑回傳指定層級的成員

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

06-3

Model.bim*

[組織架構層級2] =LOOKUPVALUE([員工名稱],[EmployeeId],PATHITEM([組織架構路徑],2,INTEGER))

到用日	性	部門名...	組織架構路徑	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
999/2/15	M	董編	1	周漢斯			
2001/2/4	M	業務部	1|106|101	周漢斯	趙德莊	王彼得	
1999/6/4	M	業務部	1|106	周漢斯	趙德莊		
2001/7/1	M	業務部	1|106|101|110	周漢斯	趙德莊	王彼得	陳蓋瑞
2001/7/1	F	業務部	1|106|101|111	周漢斯	趙德莊	王彼得	李娟娟
2001/7/1	F	業務部	1|106|101|112	周漢斯	趙德莊	王彼得	林志鈴
2001/7/1	M	業務部	1|106|101|113	周漢斯	趙德莊	王彼得	鍾漢尼
2001/7/1	M	業務部	1|106|101|114	周漢斯	趙德莊	王彼得	馬蓋先
2001/7/1	F	業務部	1|106|101|115	周漢斯	趙德莊	王彼得	吳愛愛
2001/7/1	M	業務部	1|106|116	周漢斯	趙德莊	比爾蓋	
2001/7/1	M	業務部	1|106|116|117	周漢斯	趙德莊	比爾蓋	謝阿志
2001/7/1	M	業務部	1|106|116|118	周漢斯	趙德莊	比爾蓋	李大偉
002/5/18	F	業務部	1|106|116|119	周漢斯	趙德莊	比爾蓋	陳小莉
2002/7/1	F	海外業務部	1|123|120	周漢斯	王威廉	王瑪麗	
2002/7/1	M	海外業務部	1|123|121	周漢斯	王威廉	李小龍	
002/11/1	M	業務部	1|106|116|122	周漢斯	趙德莊	比爾蓋	孫小山
003/4/15	M	海外			王威廉		
2003/7/1	M	海外			王威廉	吳保羅	
2003/7/1	M	海外業務部	1|123|125	周漢斯	王威廉	王約翰	
2005/4/6	M	業務部	1|106|126	周漢斯	趙德莊	陳志宇	
2006/3/6	F	業務部	1|106|127	周漢斯	趙德莊	王欣儀	
007/10/...	M	業務部	1|106|128	周漢斯	趙德莊	李漢宗	
010/1/30	M	業務部	1|106|129	周漢斯	趙德莊	楊紹欽	

06-3

圖6-03：回傳各層的階層成員

於是可以依此類推，來產生每個階層的計算資料行。各位可藉此清楚地理解表格模型與多維度模型父子式階層設計上的最大差異，那就是多維度模型是利用鍵值的關聯性動態生成階層結構，而表格式模型則是利用路徑函數產出每個層級的計算資料行，再將這些資料行組合成為階層，如圖6-04，**等於是利用一般維度階層的方式來做到近似父子式階層的效果**。這麼做的唯一好處就是，在資料結構上保存了父子式階層的結構特性（便於維護），而無需在資料源就事先產出每個層級所需的資料行。

06-4

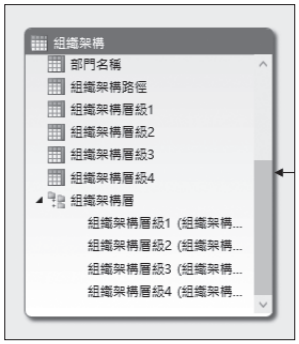


圖6-04：將路徑函數所產生的計算資料行組合成為階層

06-4

設計完階層後，我們利用Excel來檢視組織階層會得到如圖6-05的效果。各位會發現有一點怪怪的地方，那就是「王威廉」下方出現了一塊空白，這種情況就是發生了「非葉層資料（non-leaf data）」的狀況。

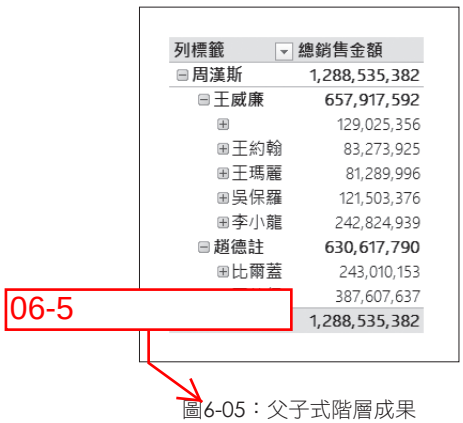


圖6-05：父子式階層成果

什麼是非葉層資料呢？一般來說，多維度分析都是在資料粒度最細的層級才會與事實資料表相連結，這個特性被稱之為「葉層資料」。也因此，絕大多數的上方階層都僅為抽象的概念。例如，事實資料表內記錄的產品編號一定是位於產品維度資料粒度最細的資料行，我們在產品階層裡面看到的產品大類（例如，智慧型手機、平板電腦...）都僅是抽象的分類概念，實際交易並不會賣出稱之為「智慧型手機」的東西，而是賣出歸屬於智慧型手機分類下的產品編號。也就是說，在傳統多維度分析裡面，只有最下方層級可能會擁有資料，上方層級是不該擁有資料的。

06-5

但在父子式階層裡面卻有可能發生上方層級擁有資料，以圖6-05來說，**主管王威廉的業績應該是由下方部屬累加上來所構成的**，但是什麼情況會出現非葉層資料呢？沒錯，就是當主管自己也帶來業績時，就有可能發生非葉層資料的情形。當王威廉自己有業績數字時，因為他的組織階層路徑是「1|123」，因此，當他解析組織架構第3層成員時，就會出現空值，所以在構成父子式階層時，會出現如圖6-05的一塊空白。

06-5

為了讓分析者容易理解這塊空白的意義，我們最好微調剛才的DAX語法。當回傳指定階層成員維空值時（DAX的空值需以ISBLANK函數判斷），就要填入該資料列自己的員工名稱。

```
DAX範例：加入填補空值的概念

=IF(ISBLANK(PATHITEM([組織架構路徑],2,INTEGER)),[員工名稱],LOOKUPVALUE([員工名稱],[EmployeeId],PATHITEM([組織架構路徑],2,INTEGER)))
```

透過這種模式填補空值後，我們再使用Excel檢視父子式階層，就能得到如圖6-06的結果。在王威廉的下方會出現一筆王威廉，而這個數字就表示是王威廉自己的業績，而非從部屬彙總上來的業績。

06-6

列標籤	總銷售金額
⊟周漢斯	1,288,535,382
⊟王威廉	657,917,592
⊞王威廉	129,025,356
⊞王約翰	83,273,925
⊞王瑪麗	81,289,996
⊞吳保羅	121,503,376
⊞李小龍	242,824,939
⊟趙德註	630,617,790
⊞比爾蓋	243,010,153
⊞王彼得	387,607,637
總計	1,288,535,382

06-6

圖6-06：填補空值後的父子式階層

表格式模型處理「非葉層資料」的方式與多維度模型是不太一樣的。在多維度模型中，可以透過「MemberWithData」來切換非葉層資料的顯示與隱藏，如圖6-07。當設定為非葉層資料顯示時，每個父成員下方一定都會出現一個同名的子成員（例如，周漢斯下方也會出現周漢斯），若設定為隱藏時，則一律隱藏。

06-7

注表格式模型中，是無法做到非葉層資料隱藏的效果，但也並非如多維度模型般，每個父成員下方一定都會出現一個同名的子成員，因為表格式模型中必須要出現非葉層的資料列，才會顯示非葉層資料（例如，圖6-06的王威廉會有，但是趙德註就沒有）。

06-6

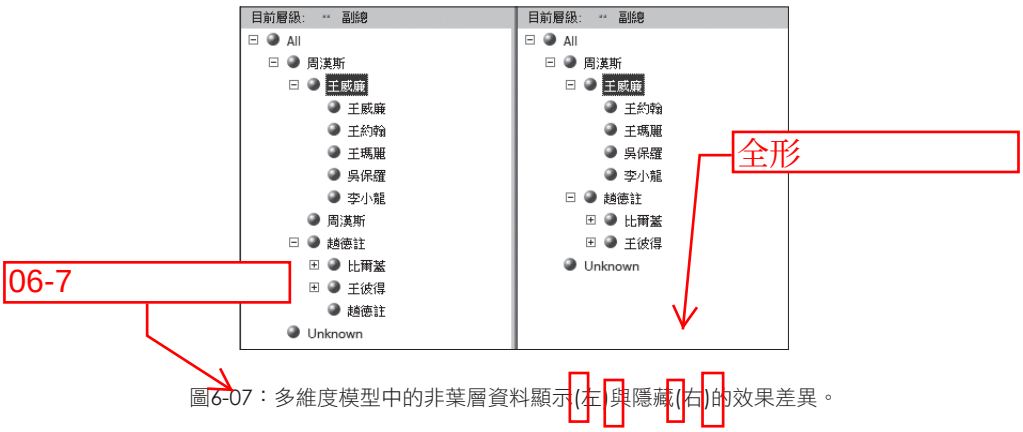


圖6-07：多維度模型中的非葉層資料顯示(左)與隱藏(右)的效果差異。

6-1-2 會計科目階層

在多維度模型中，「會計科目階層」是父子式階層中最特殊的一種。相較於一般的父子式階層，會計科目階層還多了以下的分析需求：

- 量值需要根據會計科目的種類，而有不同的彙總模式。**如果是損益表，就採取加總的方式，如果是資產負債表，則採取期末餘額的方式彙總**。
- 會計科目會根據借貸方的差異，而需要反轉運算符號。例如，營業成本在簿記時，是記錄正值，但是當要計算毛利時，**營業成本就要反轉運算符號，因此，毛利的公式變成營業收入-營業成本**。

由於會計科目計算邏輯比較複雜，因此在多維度模型中，特別提供了會計科目的維度範本，並在量值計算時提供了「ByAccount （根據會計準則）」的技術來因應這些分析需求。如圖6-08，各位可以發現在多維度模型中，透過「UnaryOperator」的設定，可讓維度圖示旁多了運算符號的標示，而量值也會依照這些運算符號進行計算。但在表格式模型中並未提供維度範本的概念。因此，我們必須搭配DAX語法來達成近似的效果。

06-8

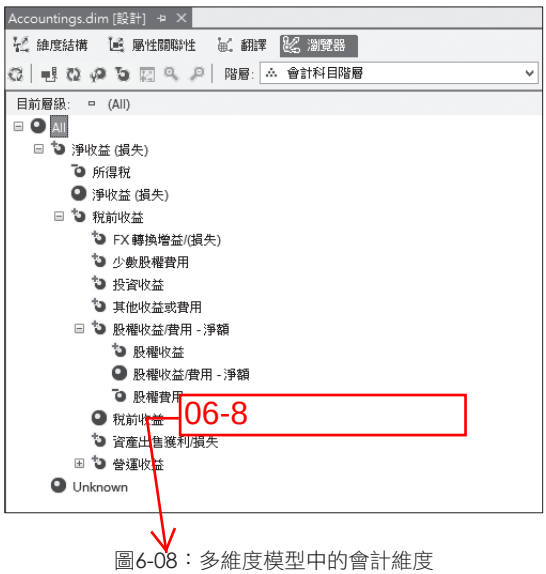


圖6-08：多維度模型中的會計維度

在本書範例「Ch06\會計科目\開始」各位可以取得已經做好初步設定的資料模型，其中包含了「日期別」、「會計科目別（Accountings）」及「會計明細（AccountingDetails）」等三個資料表。接著，我們要來示範該如何運用這三個資料表，來設計會計分析的表格式模型。

其中，最重要的資料表就是「會計科目別（Accountings）)」，為了要定義會計科目的父子式階層，資料表中提供了會計科目與父層會計科目的資料行。且為了能根據會計科目的特性來進行運算符號的轉換，該資料表中也透過「運算符號（Operator）」資料行來存放運算符號資訊，其運算符號的定義與多維度模型中的UnaryOperator相同：

運算符號	運算符號意義
+	向上加總時，該子成員資料相加後累加
-	向上加總時，該子成員資料相減後累加
*	向上加總時，父成員彙總值將乘以該子成員值（很少使用）
/	向上加總時，父成員彙總值將除以該子成員值（很少使用）
~	此子成員不向上彙總

06-9

首先，表格式模型無法像多維度模型般，自動於維度階層中顯示運算符號。既然如此，我們就只好利用DAX語法自力救濟，自行設計出帶著運算符號的會計科目名稱（計算資料行名稱為「會計科目名稱1」），如圖6-09。

```
DAX範例：產生包含運算符號的會計科目名稱

=[會計科目名稱]&" ("&[運算符號]&")"
```

會計科目	父層會計科目	會計科目名稱	運算符號	會計科目名稱1	會計科目類型	報表類型
900200		淨收益 (損失)	+	淨收益 (損失) (+)	Income	損益表
700201	900200	所得稅	-	所得稅 (-)	Expense	損益表
900300	900200	稅前收益	+	稅前收益 (+)	Income	損益表
700000	900300	股權收益/費用 - 淨額	+	股權收益/費用 - 淨額 (+)	Expense	損益表
		股權費用	-	股權費用 (-)	Expense	損益表
		股權收益	+	股權收益 (+)	Income	損益表
		資產出售獲利/損失	+	資產出售獲利/損失 (+)	Income	損益表
700004	900300	其他收益或費用	+	其他收益或費用 (+)	Income	損益表
700005	900300	FX 轉換增益/(損失)	+	FX 轉換增益/(損失) (+)	Income	損益表
700006	900300	投資收益	+	投資收益 (+)	Invest Income IC	損益表
700101	900300	少數股權費用	+	少數股權費用 (+)	Expense	損益表
900400	900300	營運收益	+	營運收益 (+)	Income	損益表

圖6-09：會計科目別資料表（資料庫表格名稱為: Accountings）

接下來將根據前一節中所介紹過的技巧，逐一產出會計科目路徑、以及各層級的內容（本範例會計科目總計有6層）：

```
DAX範例：產生會計科目路徑
=PATH([會計科目],[父層會計科目])

DAX範例：產生會計科目層級第一層
=LOOKUPVALUE([會計科目名稱1],[會計科目],PATHITEM([會計科目路徑],1,TEXT))
```

接著，再將以上透過DAX所計算出的各層資料行組合成會計科目階層，同時將會計科目明細（AccountingDetails）中的Value資料行直接加總（**金額:=SUM([Value])**），然後透過Excel來存取分析結果，可得到如圖6-10的結果。

06-10

列標籤	金額
淨收益 (損失) (+)	9,854,138
所得稅 (-)	245,000
	245,000
	245,000
	245,000
	245,000
稅前收益 (+)	9,609,138
FX 轉換增益/(損失) (+)	-
少數股權費用 (+)	-
投資收益 (+)	-
其他收益或費用 (+)	46,500
股權收益/費用 - 淨額 (+)	-
資產出售獲利/損失 (+)	-
營運收益 (+)	9,562,638
毛利 (+)	7,323,605
COGS (-)	1,073,200
營收 (+)	6,250,405
營運費用 (-)	2,239,033
總計	9,854,138

06-10

06-10

圖6-10：直接展示加總量值之結果

但圖6-10的結果是無法令人接受的。以圖上的「所得稅(-)」為例，由於這個分支總共只有兩層，因此第二層以下的部分全部都是空白。在前一節中，我們曾經教過透過補值的方式來改善，如《圖6--6：填補空值後的父子式階層》，但這種方式用在會計科目上看起來還是有點奇怪，這是首先需要在資料結構上改進之處。

06-6

第二個需改進的部分是運算符號的問題，例如，**毛利(+)應該是等於營收(+)減去COGS(-)**（COGS為業務[illegible]是直接透過加總的方式來計算量值，因此算出來的結果反而變成毛利(+)等於營收(+)加上COGS(-)。

該如何解決空白層級的問題？我們需要在會計科目資料表上產生兩個量值，分別為「目前層級」與「最深層級」，接著，就可以判斷若「目前層級」>「最深層級」，表示這是一個空白層級，那麼，就可以強制將此層級的量值變成空白。若是前端工具具有預設排除空白的功能（例如Excel），就等於可以做到隱藏空白層級的效果。

那該如何產生最深層級的量值呢？我們可以利用PATHLENGTH函數來計算該資料行的路徑深度，接下來，再透過量值計算範圍內的最深層級數。至於「目前層級」量值，也可以利用ISFILTERED函數來進行判斷，若想深入本範例提到的DAX函數，請參考本書《第7-1節：進階DAX函數》。

```
DAX範例：產生會計科目路徑層級數
=PATHLENGTH([會計科目路徑])

DAX範例：產生「最深層級」量值
最深層級:=Max([層級深度])

DAX範例：產生「目前層級」量值
目前層級:=IF(
   ISFILTERED([會計科目層級6]),6,
    IF(
       ISFILTERED([會計科目層級5]),5,
        IF(
           ISFILTERED([會計科目層級4]),4,
               IF(
                  ISFILTERED([會計科目層級3]),3,
                        IF(
                           ISFILTERED([會計科目層級2]),2,
                                 IF(
                                    ISFILTERED([會計科目層級1]),1,BLANK()
                                  ))))))
```

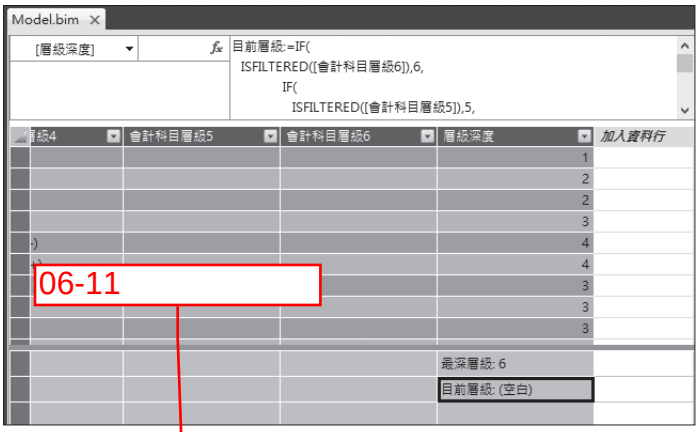


圖6-11：計算「目前層級」與「最深層級」

當計算完「目前層級」與「最深層級」量值後，只需要判斷若「目前層級」>「最深層級」，則表示為空白層級。這樣就能透過DAX函數將量值變為空白，重新部署後可看到如圖6-12的結果。此時像是所得稅(-)、營收(+)、COGS(-)等階層成員就不再可展開（因為下方的層級都被自動篩選空白給濾掉了）。如此一來，就先解決了第一個結構上呈現的問題。

06-12

```
DAX範例：隱藏空白層級的量值
金額1:=IF([目前層級]>[最深層級],BLANK(),SUM([Value]))
```

列標籤	金額1
⊟淨收益 (損失) (+)	9,854,138
所得稅 (-)	245,000
⊟稅前收益 (+)	9,609,138
⊞FX 轉換增益/(損失) (+)	-
⊞少數股權費用 (+)	-
⊞投資收益 (+)	-
⊞其他收益或費用 (+)	46,500
⊞股權收益/費用 - 淨額 (+)	-
⊞資產出售獲利/損失 (+)	-
⊟營運收益 (+)	9,562,638
⊟毛利 (+)	7,323,605
GS (-)	1,073,200
益 (+)	6,250,405
⊞營運費用 (-)	2,239,033
總計	9,854,138

06-12

圖6-12：隱藏空白層級後的會計值

06-13

只解決了階層呈現的問題還不夠，該如何根據指派的運算符號來計算量值才是真正棘手的問題。我們透過圖6-13來解釋該如何解決，以「薪資與報酬」為例，這個科目是第4層的科目，它本身所帶的運算符號是正號。當我們從它的上一層（第3層）檢視時，由於第3層科目「營運費用」的運算符號為負，所以負正得負。從第3層的角度看「薪資與報酬」科目時，整體來看的運算符號仍舊是負號。那麼當我們從第2層以及第1層來看時，由於這兩層都是正號，因此第2層是正負得負，第1層也是正負得負。即便以第1層的角度來看「薪資與報酬」科目時，整體的運算符號也依舊是負號的。

圖6-13：正負號反轉邏輯示意圖

06-14

為了實作此邏輯，筆者於「會計科目別」資料表產生「運算符號1」的計算資料行，將原本文字的運算符號轉換為數值（1/ -1）。接著，再透過DAX來計算從第6層檢視各個科目時的運算符號，各位可以參考圖6-14，可以發現透過以下的DAX範例，僅有第6層的科目才會回傳運算符號值。

接著，我們從第5層來檢視運算符號。第5層的運算符號等於第5層本身的運算符號乘以第6層的運算符號，各位可參考以下的DAX範例，但由於只要第5層有值的科目，理應於第6層也都應該有值，因而在此，透過IF函數將空白值補為1。以此類推，可逐一計算產生至第4、3、2、1層的運算符號，計算後的結果請參考圖6-14。

06-14

```
DAX範例：運算符號1 （「會計科目別」資料表的計算資料行）
=IF([運算符號]="+",1,IF([運算符號]="-",-1,BLANK()))

DAX範例：運算符號層級6（「會計科目別」資料表的計算資料行）
=LOOKUPVALUE([運算符號1],[會計科目],PATHITEM([會計科目路徑],6,TEXT))

DAX範例：運算符號層級5（「會計科目別」資料表的計算資料行）
=LOOKUPVALUE([運算符號1],[會計科目],PATHITEM([會計科目路徑],5,TEXT))*IF(ISBLANK([運算符號層級6]),1,[運算符號層級6])
```

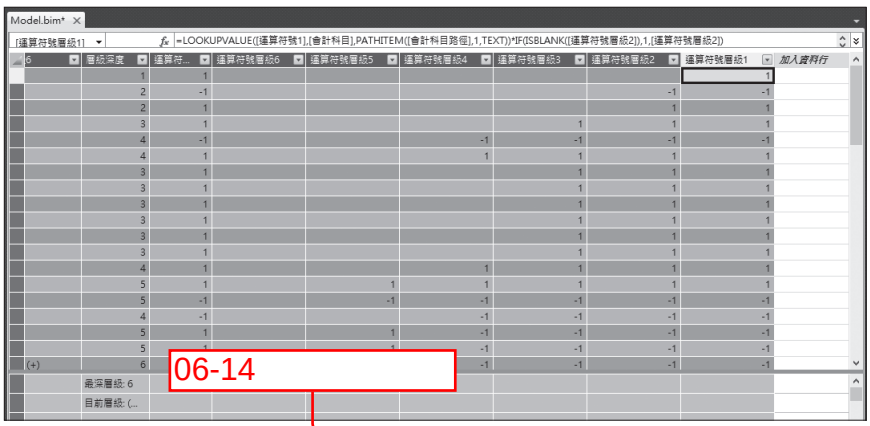


圖6-14：計算各層級檢視時的正負號

接下來，就需根據目前的層數是第幾層，然後套用剛才產生的各層運算符號來處理科目的彙總值。舉例來說，如果目前層級為第1層，那麼所有科目皆需根據

「運算符號層級1」來進行運算。將「運算符號層級1」為1之科目加總減去「運算符號層級1」為-1之科目，如此一來，就能得到正確的計算結果。至於要做到僅將「運算符號層級1」為1之科目加總，只需透過CALCULATE函數就能做到。DAX範例如下：

DAX範例：處理根據運算符號判斷計算模式

```
金額2:=
    IF ([目前層級]<=[最深層級],
IF ([目前層級]=1,
         CALCULATE (SUM ('會計明細'[Value]), '會計科目別'[運算符號層級1] = 1)
      - CALCULATE (SUM ('會計明細'[Value]), '會計科目別'[運算符號層級1] = -1)
      ,
        IF ([目前層級]=2,
          CALCULATE (SUM ('會計明細'[Value]), '會計科目別'[運算符號層級2] = 1)
        - CALCULATE (SUM ('會計明細'[Value]), '會計科目別'[運算符號層級2] = -1)
        ,
          IF ([目前層級]=3,
            CALCULATE (SUM ('會計明細'[Value]), '會計科目別'[運算符號層級3] = 1)
          - CALCULATE (SUM ('會計明細'[Value]), '會計科目別'[運算符號層級3] = -1)
          ,
            IF ([目前層級]=4,
              CALCULATE (SUM ('會計明細'[Value]), '會計科目別'[運算符號層級4] = 1)
            - CALCULATE (SUM ('會計明細'[Value]), '會計科目別'[運算符號層級4] = -1)
            ,
              IF ([目前層級]=5,
                 CALCULATE (SUM ('會計明細'[Value]), '會計科目別'[運算符號層級5] = 1)
               - CALCULATE (SUM ('會計明細'[Value]), '會計科目別'[運算符號層級5] = -1)
                                  ,
                                           IF ([目前層級]=6,
                                             CALCULATE (SUM ('會計明細'[Value]), '會
計科目別'[運算符號層級6] = 1)
                                           - CALCULATE (SUM ('會計明細'[Value]), '會
計科目別'[運算符號層級6] = -1)
                                           ,
                                           BLANK()
                                           )
                                  )
                         )
                )
        )
   )

  ,BLANK())
```

06-15

我們將原先的計算結果與透過DAX判斷運算符號的結果並排在一起比較，如圖6-15（金額1是原本的彙總模式，金額2是處理了運算符號），就能夠一目了然其中的差異。各位會發現被標上「(-)」的會計科目，其量值都自動變為負值，也因此毛利的計算結果就能正確地更改為營收(+)減去COGS(-)。

列標籤	金額1	金額2
⊟淨收益 (損失) (+)	9,854,138	2,739,672
所得稅 (-)	245,000	-245,000
⊟稅前收益 (+)	9,609,138	2,984,672
FX 轉換增益/(損失) (+)	0	0
少數股權費用 (+)	0	0
投資收益 (+)	0	0
其他收益或費用 (+)	46,500	46,500
⊞股權收益/費用 - 淨額 (+)	0	0
資產出售獲利/損失 (+)	0	0
⊟營運收益 (+)	9,562,638	2,938,172
⊟毛利 (+)	7,323,605	5,177,205
COGS (-)	1,073,200	-1,073,200
營收 (+)	6,250,405	6,250,405
⊟營運費用 (-)	2,239,033	-2,239,033
⊞公共事業費用 (+)	35,170	35,170
⊞行銷 (+)	51,560	51,560
⊞旅 (+)	5,665	5,665
⊞[illegible]費用 (+)	134,128	134,128
⊞辦公室成本 (+)	99,480	99,480
⊞薪資與報酬 (+)	1,913,030	1,913,030
總計	9,854,138	2,739,672

06-15

圖6-15：根據運算符號判斷計算模式前後之結果

但現在這樣仍不夠完美，雖然透過上述的方式已能正確地計算出會計科目金額。但就正確的會計科目簿記準則來看，「營運費用(-)」科目雖然計算時，要是以負值來計算，但在報表呈現時，則必須顯示為正值才對。那麼該如何在顯示時為正值，計算時卻當成負值來處理呢？

問題不大，只需判斷目前層級該科目的運算符號即可。如果是負值，就表示應該要將目前的金額乘上-1，來解決上述的問題。所以在以下的DAX中，首先於「會計明細」資料表中產生「目前運算符號」量值，然後將「金額2」乘上「目前運算符號」，就能算出正確的金額了。

DAX範例：建立判斷目前運算符號的量值

```
目前運算符號:=
IF ([目前層級]<=[最深層級],
  IF ([目前層級]=1,
     MAX('會計科目別'[運算符號層級1]),
        IF ([目前層級]=2,
          MAX('會計科目別'[運算符號層級2]) ,
                IF ([目前層級]=3,
                  MAX('會計科目別'[運算符號層級3]),
                        IF ([目前層級]=4,
                         MAX('會計科目別'[運算符號層級4]),
                                IF ([目前層級]=5,
                                  MAX('會計科目別'[運算符號層級5]) ,
                                          IF ([目前層級]=6,
                                                  MAX('會計科目別'[運算符號層級6]),
                                                     1)))))) ,BLANK())
```

06-16

DAX範例：修正正負號顯示之問題

```
金額3:=[金額2]*[目前運算符號]
```

參考圖6-16，我們將3種算法一字排開，各位可以發現「金額3」的總金額與「金額2」是相同的。但原本運算符號為負號的科目在「金額2」顯示為負值，但在「金額3」中，卻能正確地以正值的方式顯示，如此就完美地達到會計科目分析的基本需求了。

列標籤	金額1	金額2	金額3	目前運算符號
⊟淨收益 (損失) (+)	9,854,138	2,739,672	2,739,672	1
所得稅 (-)	245,000	-245,000	245,000	-1
⊟稅前收益 (+)	9,609,138	2,984,672	2,984,672	1
FX 轉換增益/(損失) (+)	0	0	0	1
少數股權費用 (+)	0	0	0	1
投資收益 (+)	0	0	0	1
其他收益或費用 (+)	46,500	46,500	46,500	1
⊞股權收益/費用 - 淨額 (+)	0	0	0	1
資產出售獲利/損失 (+)	0	0	0	1
⊟營運收益 (+)	9,562,638	2,938,172	2,938,172	1
⊟毛利 (+)	7,323,605	5,177,205	5,177,205	1
COGS (-)	1,073,200	-1,073,200	1,073,200	-1
營收 (+)	6,250,405	6,250,405	6,250,405	1
⊟營運費用 (-)	2,239,033	-2,239,033	2,239,033	-1
⊞公共事業費用 (+)	35,170	35,170	35,170	1
⊞行銷 (+)	51,560	51,560	51,560	1
[illegible]	5,665	5,665	5,665	1
[illegible]	134,128	134,128	134,128	1
⊞辦公室成本 (+)	99,480	99,480	99,480	1
⊞薪資與報酬 (+)	1,913,030	1,913,030	1,913,030	1
總計	9,854,138	2,739,672	2,739,672	1

06-16

圖6-16：處理(-)項正負號顯示問題

不過，在此還是要提醒各位，表格式模型的定位為精簡版的商業智慧導入技術，筆者雖然在此透過DAX實作出會計分析模型，但若想要做到更複雜的會計財務分析，還是建議您採用多維度模型會是較好的方法。

06-02 進階商業分析設計

接下來，我們將介紹如何透過表格式模型，來實作一些較複雜的商業智慧分析情境。

6-2-1 多對多維度關係設計

多對多維度關係設計一直是商業智慧設計上難度非常高的一塊，它通常用來處理「分項加總不等於總計」的狀況。而會造成這種情況的因素有兩種：

- 去重複計數（DistinctCount）。
- 複選題或共用資訊（例如，處理共用帳號、共用擔保品…）。

「去重複計數」對於多維度分析來說，是非常嚴重的問題。因為多維度分析的本質是根據預先彙總，但去重複計數卻違反了這個前提，因此，變成分析者必須要採用較複雜的多對多維度技術來解決。但對於表格式模型來說，反正所有的查詢都是在記憶體中即時算出來的，再加上表格式模型有針對資料行內容，預先編制索引，因而去重複計數其實並不複雜，只需在設計量值時，使用預設的「自動加總」功能，就能夠將彙總模式設定為DistinctCount，或是透過DAX的DistinctCount函數來撰寫去重複計數量值邏輯。

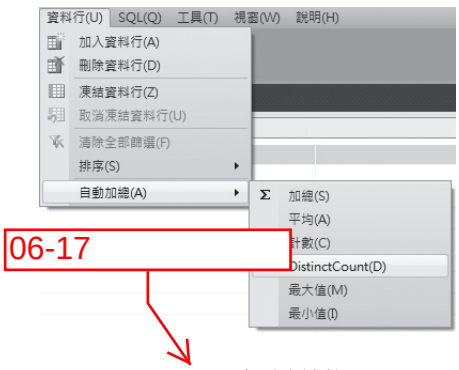


圖6-17：去重複計數

06-18

但隨著模型中交錯的邏輯更複雜時，光使用DistinctCount函數還是不夠。接下來，我們將透過一個電信業的範例，來向各位說明常見的多對多維度情境，同時也讓客戶看到如何透過表格式模型，來簡化這部分的開發。各位可以參考本書範例「Ch06\多對多\開始」，在範例資料模型中，共計有以下資料表（如圖6-17）：

- 電信帳單明細（TelecomBillings）
- 門號別（TelecomMSISDN）
- 客戶別（TelecomCustomers）
- 費率別（TelecomRatePlans）
- 帳務日期別（TelecomDate）

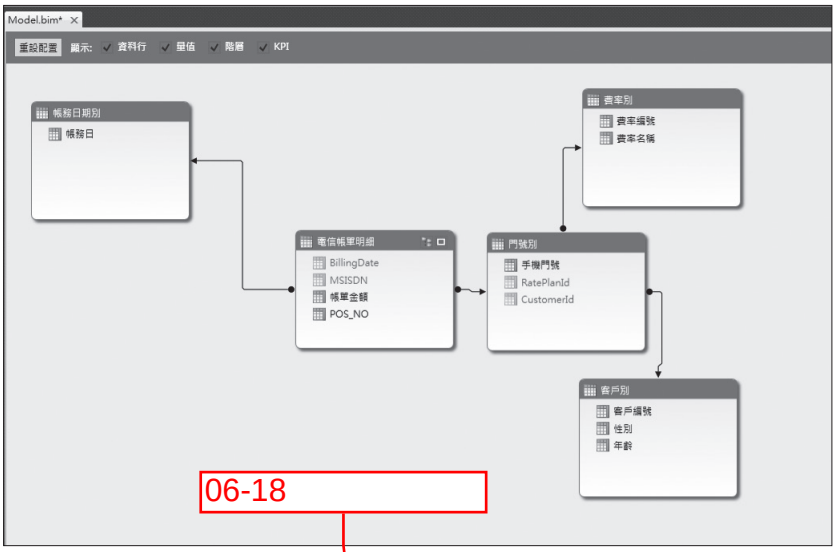


圖6-18：電信多對多範例

什麼是「多對多維度」？在傳統維度資料表與事實資料表的對應關係中，大多屬於**多（事實資料表）對一（維度資料表）的關係**。在處理資料彙總時，是以維度資料表為主索引鍵，連結事實資料表的外部索引鍵。因此，每一筆事實資料表裡的資料，都只會對到單一的維度成員，以產生彙總計算結果。

但很多實務的情況卻非如此。舉例來說，在電信業裡面，一個客戶可能會持有多組門號，對信用卡來說，一個客戶可能會持有多張卡，而客戶也可以在多家銀行

分行裡面開戶。這種情況變成不再是多對一的對應。舉例來說，一個客戶有多支手機門號，而每個手機門號可能會對應到不同的費率。當我們希望將「費率」維度與「顧客數量」量值連結時，**會發生一個客戶可能會同時出現在不同費率的狀況**，這種情形我們就稱之為多對多維度。

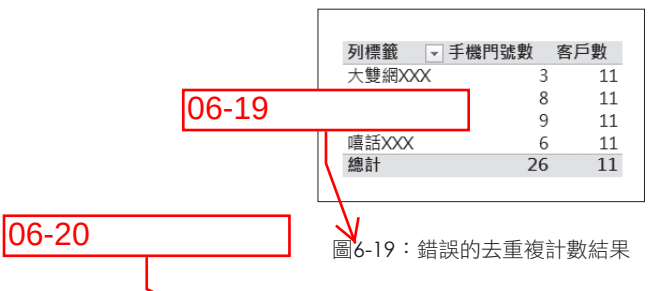
多對多維度能夠解決資料倉儲中普遍的分析難題。對許多的企業來說，帳號跟客戶大多是不一致的。像是方才提過的電信業者裡，一個客戶可能會有多個門號，而每支門號都會有個別的費率、開通日、開通通路等資訊。對於企業來說，除了統計帳戶數目之外，有時候更需要計算企業的客戶歸戶數（Counterparty），但可能會有客戶擁有的不同門號屬於不同的費率。因此，同一個客戶會出現在不同費率，造成各費率的客戶數加總後，會遠多於總客戶數（常見的複選題或去重複計數問題）。

首先以最直觀的方式，在模型內加入以下量值：

```
DAX範例：計算去重複客戶數
客戶數:=DISTINCTCOUNT([客戶編號])

DAX範例：計算去重複門號數
手機門號數:=DISTINCTCOUNT([手機門號])
```

但如果透過Excel連結至剛才設計的表格式模型，我們會發現資料會計算錯誤，其中「客戶數」量值會全部變成11人（與總數相同）。



列標籤	手機門號數	客戶數
大雙網XXX	3	11
	8	11
	9	11
嘻話XXX	6	11
總計	26	11

圖6-19：錯誤的去重複計數結果

我們可以透過圖6-20來說明錯誤的原因。如果要計算各個費率下的客戶數，那麼「費率別」連結至「客戶別」之間還隔著「門號別」的資料表。由於目前表格式模型只支援**雪花狀維度**，這種情況會與「產品大類別」透過「產品別」連結至「銷售明細」的邏輯相似，前提是維度資料表連結至事實資料表必須保持「一對多」的關係。但在此的門號別與客戶別卻處於相反的關係。因為一個客戶可以擁

有多個門號，反而是變成「多對一」的關係，因此，表格式模型無法正確地處理對應關係，於是它只好回傳整個資料表。才會造成全部算出來的客戶數都等於總數的問題。

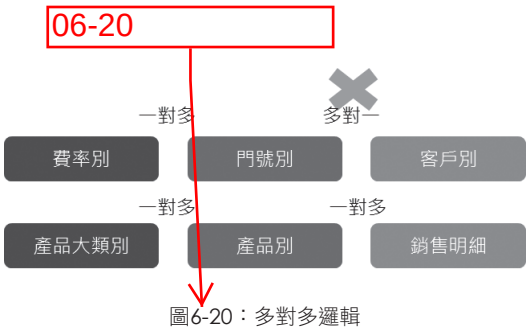


圖6-20：多對多邏輯

要解決這個問題並不複雜，只需要簡化維度結構，避開「多對一」的問題就可以了。做法是只在中介資料表「門號別」上，利用DAX的Related函數產生「門號所有者客戶編號」計算資料行（當然，這個資料行不會讓分析者直接存取，因而可設為隱藏），然後再透過DistinctCount函數計算「門號所有者客戶編號」數量，如圖6-21。由於少掉了「多對一」的關係，自然就能避開表格式模型不支援多對多維度的問題。

```
DAX範例：在門號別資料表上產生「門號所有者客戶編號」計算資料行
=RELATED('客戶別'[客戶編號])

DAX範例：計算去重複門號數
多對多客戶數:=DISTINCTCOUNT([門號所有者客戶編號])
```

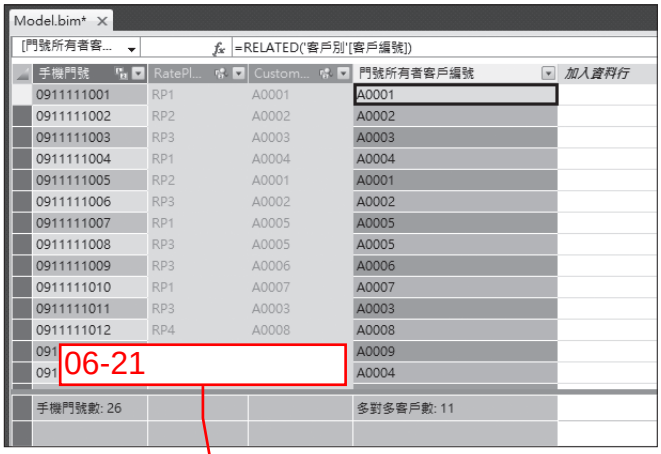
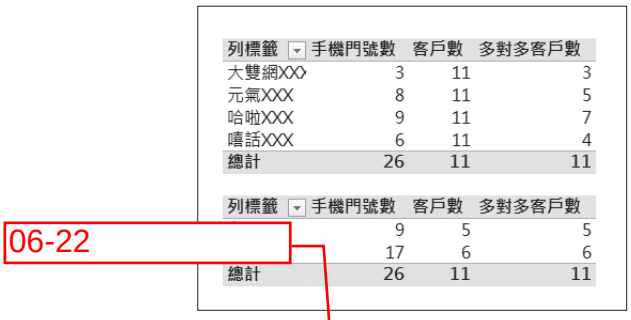


圖6-21：利用Related簡化模型

06-22

我們可以比較一下利用這種方式所算出來的結果。如圖6-22，發現算出來的是我們預期的結果，可以正確顯示出各個資費的客戶數，另外還會發現各個資費的客戶數加總（19人）是多於總客戶數（11人）的，這也符合我們對於多對多維度的期待。至於檢視一般維度時（性別），兩種客戶數都能呈現出正確的數字，且此時分享加總會等於總數。



列標籤	手機門號數	客戶數	多對多客戶數
大雙網XXX	3	11	3
元氣XXX	8	11	5
哈啦XXX	9	11	7
嘻話XXX	6	11	4
總計	26	11	11

列標籤	手機門號數	客戶數	多對多客戶數
	9	5	5
	17	6	6
總計	26	11	11

圖6-22：正確的多對多量值

06-23

筆者認為，雖然在技術文件上說明表格式模型不支援多對多維度設計，但並不表示表格式模型的功能比較少，而是以它的架構來說，根本不用那麼複雜的結構就能夠處理。不信的話，你們可以參考圖6-23，要達到相同效果，多維度模型必須要設定多組資料表間關係才能做到，如此看來，表格式模型處理多對多維度反而比較明瞭易懂。

圖6-23：多維度模型中的多對多維度設定

06-23

6-2-2 時間智慧分析

在商業智慧分析中，時間智慧是非常重要的議題，這也是為何多維度計數當初大受歡迎的主因之一。傳統要透過T-SQL查詢來取得時間智慧資訊並不好分辨，以

資料格為計算基礎的多維度模型，能大幅簡化時間智慧分析的複雜度。但相對來說，多維度模型技術架構上雖然很容易做到時間智慧分析，但由於必須透過高難度的MDX語法，因此在推廣上，還是具一定的困難度。在此筆者將會示範一系列如何在表格式模型中，透過DAX來實作時間智慧的技巧，即使各位不了解DAX全貌，也保證能夠輕鬆上手。

其實表格式模型的DAX語法中，特地為了時間智慧，設計出一系列的相關函數。由於它的結構類似Excel函數，因此學習起來的困難度遠低於MDX。首先，來看如何在表格式模型中做出YTD（Year-To-Date，年度累計）、QTD（季度累計）以及MTD（月度累計）的分析。

```
DAX範例：計算年度累計金額
YTD總銷售金額:=TotalYTD([總銷售金額],'銷售日別'[資料日])

DAX範例：計算季度累計金額
QTD總銷售金額:=TotalQTD([總銷售金額],'銷售日別'[資料日])

DAX範例：計算月度累計金額
MTD總銷售金額:=TotalMTD([總銷售金額],'銷售日別'[資料日])
```

06-24

只需透過TotalYTD等函數，並傳入量值與日期資料行，就能夠輕鬆實作時間累計分析，如圖6-24。

列標籤	總銷售金額	YTD總銷售金額	QTD總銷售金額	MTD總銷售金額
⊞2010	18,537,170	18,537,170	14,850,060	5,089,880
⊞2011	166,490,529	166,490,529	76,381,880	30,472,233
⊞2012	487,712,546	487,712,546	127,444,279	36,783,530
⊞2013	321,268,789	321,268,789	67,440,935	24,824,769
⊟2014	303,930,173	303,930,173	52,028,785	10,689,080
⊞2014/1/1	26,964,704	26,964,704	26,964,704	26,964,704
⊞2014/2/1	28,895,340	55,860,044	55,860,044	28,895,340
⊞2014/3/1	30,590,574	86,450,618	86,450,618	30,590,574
⊞2014/4/1	27,778,909	114,229,527	27,778,909	27,778,909
⊞2014/5/1	26,102,723	140,332,250	53,881,632	26,102,723
⊞2014/6/1	24,451,294	164,783,544	78,332,926	24,451,294
⊞2014/7/1	26,891,377	191,674,921	26,891,377	26,891,377
⊞2014/8/1	31,049,827	222,724,748	57,941,204	31,049,827
⊞2014/9/1	29,176,640	251,901,388	87,117,844	29,176,640
⊞2014/10/1	23,363,825	275,265,213	23,363,825	23,363,825
⊞2014/11/1	17,975,880	293,241,093	41,339,705	17,975,880
⊞2014/12/1	10,689,080	303,930,173	52,028,785	10,689,080
總計	1,297,939,207	303,930,173	52,028,785	10,689,080

圖6-24：時間累計分析

06-24

另一個常見的時間智慧分析情境就是成長率比較。常見的成長率比較情境是與前月相比，或者是與去年同期相比。原則上都是利用Calculate函數來重新根據只並時間範圍計算數值。至於要檢視哪個時間點，就可以透過DateAdd函數來評估了。由於去年同期比較這個情境太過常見，DAX中還特地為它設計了SamePeriodLastYear函數以簡化計算。

```
DAX範例：計算去年同期銷售金額(Previous Year)
PY總銷售金額:=Calculate([總銷售金額],SamePeriodLastYear('銷售日別'[資料日]))
DAX範例：計算去年同期銷售金額(Previous Year)
PY總銷售金額:=Calculate([總銷售金額],DateAdd('銷售日別'[資料日],-1,Year))
06-25
DAX範例：計算前月銷售金額(Previous Month)
PM總銷售金額:=Calculate([總銷售金額],DateAdd('銷售日別'[資料日],-1, Month))
```

計算結果如圖6-25，各位可以清楚地觀察期間比較分析的結果。

列標籤	總銷售金額	PY總銷售金額	PM總銷售金額
⊞2010	18,537,170		13,447,290
⊞2011	166,490,529	18,537,170	141,108,176
⊞2012	487,712,546	166,490,529	481,401,249
⊟2013	321,268,789	487,712,546	333,227,550
⊞2013/1/1	32,619,213	38,971,835	36,783,530
⊞2013/2/1	29,090,840	38,175,831	32,619,213
⊞2013/3/1	27,848,434	34,444,234	29,090,840
⊞2013/4/1	29,753,727	32,678,094	27,848,434
⊞2013/5/1	27,185,747	36,739,131	29,753,727
⊞2013/6/1	28,780,637	41,519,401	27,185,747
⊞2013/7/1	26,773,276	45,204,337	28,780,637
⊞2013/8/1	26,266,407	46,566,397	26,773,276
⊞2013/9/1	25,509,573	45,969,007	26,266,407
⊞2013/10/1	22,524,531	45,068,734	25,509,573
		45,592,015	22,524,531
06-25		36,783,530	20,091,635
⊞2014	303,930,173	321,268,789	318,065,862
總計	1,297,939,207	994,009,034	1,287,250,127

圖6-25：去年同期與前月比較

既然有了去年同期銷售金額，自然就可以和今年的銷售金額比較成長率。但要注意的是，在計算成長率時，還是得考慮分母為空值之問題，因此，我們可以透過以下的公式來計算去年同期成長率：

```
DAX範例：計算銷售金額去年同期成長率
銷售金額去年同期成長率:=if(IsBlank([PY總銷售金額]),Blank(),([總銷售金額]-[PY總銷售金額])/[PY總銷售金額])
```

列標籤	總銷售金額	PY總銷售金額	銷售金額去年同期成長率
⊞2010	18,537,170		
⊞2011	166,490,529	18,537,170	798.14%
⊞2012	487,712,546	166,490,529	192.94%
⊟2013	321,268,789	487,712,546	-34.13%
⊞2013/1/1	32,619,213	38,971,835	-16.30%
⊞2013/2/1	29,090,840	38,175,831	-23.80%
⊞2013/3/1	27,848,434	34,444,234	-19.15%
⊞2013/4/1	29,753,727	32,678,094	-8.95%
⊞2013/5/1	27,185,747	36,739,131	-26.00%
⊞2013/6/1	28,780,637	41,519,401	-30.68%
⊞2013/7/1	26,773,276	45,204,337	-40.77%
⊞2013/8/1	26,266,407	46,566,397	-43.59%
⊞2013/9/1	25,509,573	45,969,007	-44.51%
⊞2013/10/1	22,524,531	45,068,734	-50.02%
⊞2013/11/1	20,091,635	45,592,015	-55.93%
⊞	06-26	530	-32.51%
⊞20		789	-5.40%
總計	1,297,939,207	994,009,034	30.58%

圖6-26：去年同期成長率

06-03 檢視方塊與資料分割

6-3-1 檢視方塊

當表格式模型中包含越來越多的資料表，高度的複雜性反而會讓分析人員感到困惑，難以找到他們想要分析的內容。此時，就可以利用檢視方塊（Perspectives）來簡化表格式模型的結構。

檢視方塊可視為表格式模型的檢視表（View）。我們可以根據分析主題定義檢視方塊，並於檢視方塊中納入與此主題相關的資料表。如此一來，當分析人員切換至指定檢視方塊時，就能方便地找尋所需的分析資訊。

不過，要特別注意的是，檢視方塊與「安全性」設定無關，它僅是一種簡化模型的便利機制，並不能用來限定使用者能看到哪些資料表。

在表格式模型中設計檢視方塊，必須點選工具列的「模型」→「檢視方塊」→「建立與管理」。此時，會開啟如圖6-28的檢視方塊設計工具。

06-28

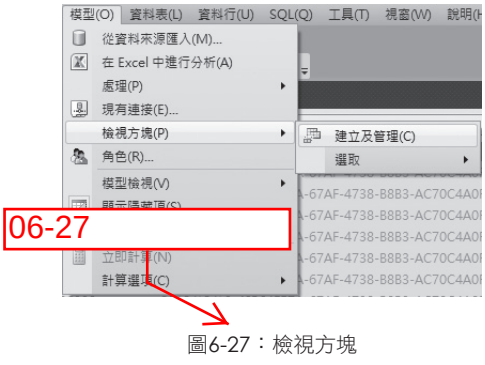


圖6-27：檢視方塊

若想要新增檢視方塊，只需點選左上方的「新增檢視方塊」按鈕，然後勾選要納入此模型中的資料表或資料行即可。點選檢視方塊名稱，即會顯示刪除或重新命名的按鈕。

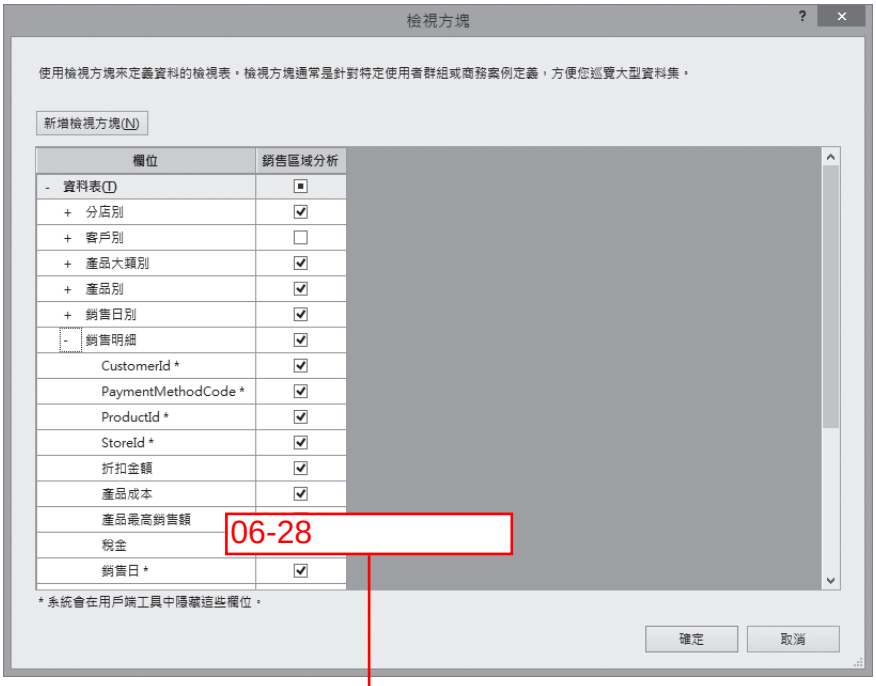


圖6-28：設定檢視方塊

帶設定完檢視方塊且部署至伺服器後，當使用者要建立與此資料庫的連接時，即可於畫面中切換至想要分析之檢視方塊，如圖6-29，就能免去於茫茫大海中找尋想要資料表的痛苦了。

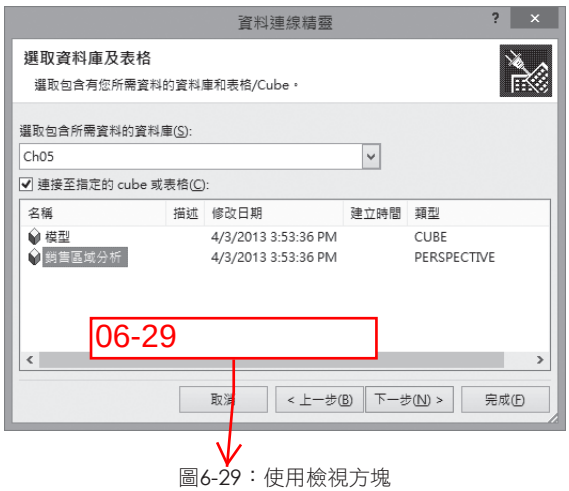


圖6-29：使用檢視方塊

6-3-2 資料分割

在多維度模型中，為了解決因資料量過大而造成的效能議題時，經常會使用資料分割的技術，將背後的資料表切割為不同範圍，以提升處理與查詢時的效能。

在表格式模型中雖然也有資料分割機制，但意義卻與多維度模型大不相同。其差異之處在於：

- 表格式模型中，資料分割僅用來定義資料模型處理的範圍，與效能無關。
- 多維度模型中，僅有事實資料表可以設定資料分割，但是在表格式模型中，任何資料表都可以設定資料分割。

「**與效能無關**」這個觀念是必須要在此強調的，因為表格式模型並非以資料列的方式來進行運算，因此，將資料表分割是無助於效能提升的。若是要改善表格式模型的效能，應該要從資料型別、結構下手會比較有意義（參考本書《第10-2節：BISM表格式模型記憶體使用最佳化》）。那既然無關效能，那麼設定資料分割的好處是什麼呢？

- **減少日常處理時間**：資料分割是表格式模型處理的最小單位。當大型資料表劃分為數個資料分割時，僅需要更新該資料分割範圍內的資料，可有效的縮短日常處理時間。不過要注意的是，過度細切資料分割未必能夠獲得縮短處理時間的好處，因為一個資料表內的資料分割是不允許同時執行的，只能夠透過依序執行的方式，因此，若是過分細切資料分割，反而有可能拉長處理時間。

- **可以彈性定義分析資料範圍**：若企業希望設定資料模型中僅保留最近n年的歷史資料，那麼，透過資料分割設定就可以方便地透過刪除就資料分割的方式，來達成資料管理的目的。
- **整合多個資料來源**：資料分割允許每個資料分割來自於不同的資料表。當資料來自於相同結構的多個資料表時（例如，每個月存一個資料表）06-30
資料分割將資料整合。

若要設計資料分割，需點選工具列的「資料表」→「資料分割」開啟如圖6-30的資料分割設計工具。上方的「資料表」下拉選單可用來切換要設計資料分割的資料表。您可以透過「新增」的方式來新增資料分割定義，或者是點選現有的資料分割後，按下「複製」，以複製現行定義後修改。

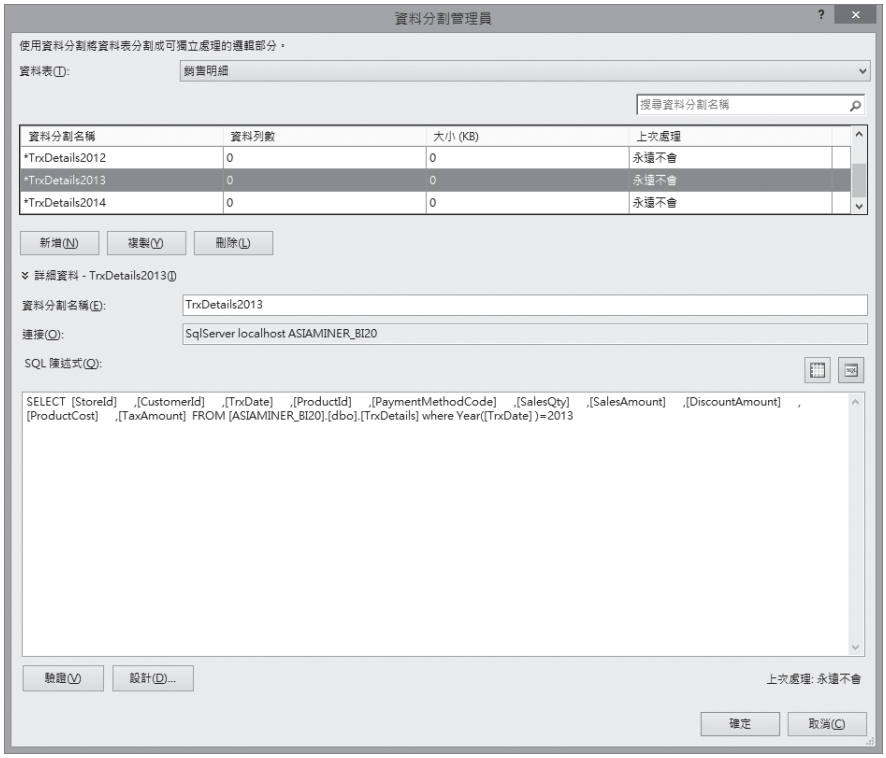


圖6-30 ：資料分割設計

06-30

06-30

以圖6-30為例，筆者是依照資料年度來設計資料分割，僅需在「SQL陳述式」中加入對應的Where條件（若源頭資料是一個年度存成一個資料表，也可以在此指定至不同資料表 06-31 定。

當資料分割設計完成後，可點選工具列的「模型」→「處理」→「處理資料分割」後，即會彈出如圖6-31的處理資料分割對話方塊。此時，只需勾選育處理之資料分割以及想要的處理模式，即可更新資料分割內容。

圖6-31：處理資料分割

至於日後的資料分割處理與管理，各位可以參考本書《第9章：部署、處理與安全性》之內容。◆◆

06-31