

# Notions avancées en MDX

Chapitre 5

Business Intelligence

# Résumé sur MDX

- Instruction MDX simple :

```
SELECT axis [,axis] FROM cube WHERE slicer [,slicer]
```

- La clause SELECT est utilisée pour définir les **dimensions des axes** et la clause WHERE pour spécifier les **dimension de filtrage**
- SELECT : si le cube est vu comme une structure à n dimensions, cette clause spécifie les arêtes du cube à retourner :

```
SELECT Set1 ON axis_name,  
      Set2 ON axis_name,  
      Set3 ON axis_name
```

  - où axis\_name peut être COLUMNS ou ROWS, 0, 1, ...
- WHERE : Les dimensions de filtrage contiennent les seuls membres avec lesquels le cube est filtré (slicé)

# Résumé sur MDX

- Un membre est un élément spécifique de donnée dans une dimension :

`[Time].[2012]`

`[Customers].[All Customers].[Mexico].[Mexico]`

`[Product].[All Products].[Drink]`

- Un tuple est une collection de membres de différentes dimensions :

`([Time].[2012], [Product].[All Products].[Drink])`

`(2012, Drink)`

`(2012, [Customers].[All Customers].[Mexico].[Mexico])`

- Un Set est une collection de tuples utilisée pour construire un axe :

`{[Time].[ 2012], [Time].[ 2013], [Time].[ 2014]}`

`{(2012, Drink), (2013, Drink)}`

# Partie 1 : Gestion des membres et des tuples

Points traités :

- Emboitement (Nest) de tuples

- Membres calculés

- Membres NULL et Cellules EMPTY

# Emboitement (Nest) de tuples

- Les axes peuvent contenir des membres ou des tuples
  - Exemple : On veut voir les ventes de 'computers' et 'printers' en Europe et en Asie sur les années 2012 et 2013. On peut le faire avec une requête MDX avec 3 axes :

```
SELECT
    { Continent.[Europe], Continent.[Asia] } ON AXIS(0),
    { Product.[Computers], Product.[Printers] } ON AXIS(1),
    { Years.[2012], Years.[2013] } ON AXIS(2)
FROM Sales
```

- Le résultat de cette requête n'est pas une table à 2 dimensions, mais un cube à 3 dimensions, **difficile à interpréter**

# Emboitement (Nest) de tuples

- Solution : Réécrire la requête en considérant les 3 dimensions, mais en utilisant seulement 2 axes, avec une dimension emboîtée dans une autre
  - La dimension Product sera emboîtée (nested) dans la dimension Regions conduisant aux 4 combinaisons des membres de dimensions {Europe, Asie} et {Computers, Printers}
  - Chaque combinaison de membres de dimensions différentes est un Tuple :

SELECT

```
{ Year.[2012], Year.[2013] } ON COLUMNS,  
{ ( Continent.Europe, Product.Computers ),  
  ( Continent.Europe, Product.Printers ),  
  ( Continent.Asia, Product.Computers ),  
  ( Continent.Asia, Product.Printers ) } ON ROWS
```

FROM Sales

		2012	2013
Europe	Computers		
	Printers		
Asia	Computers		
	Printers		

# Membres calculés

- MDX permet d'étendre le cube en définissant d'autres membres de dimensions, qui sont calculés à partir de membres existants
  - La syntaxe pour les membres calculés utilise la clause WITH avant l'instruction SELECT :

```
WITH MEMBER parent_dim.new_name AS 'expression'
```

- Les membres calculés sont traités de la même manière que des membres ordinaires, ils peuvent donc être utilisés :
  - dans la définition d'axes
  - dans un filtre WHERE

# Membres calculés

- Exemple : Trouver les profits (bénéfice) des produits au cours des années :

```
WITH  
    MEMBER Measures.Profit AS 'Measures.Sales - Measures.Cost'  
SELECT  
    Products.MEMBERS ON COLUMNS,  
    Year.MEMBERS ON ROWS  
FROM Sales  
WHERE ( Measures.Profit )
```

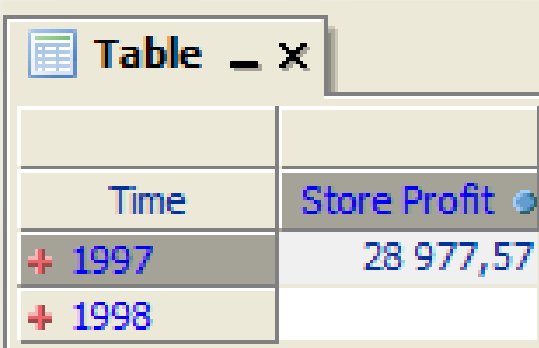
- Le parent du nouveau membre est la dimension « Measures »
- Le nom du nouveau membre est « Profit »
- L'expression de calcul est « Profit = Sales – Cost »



# Membres calculés

- Exemple:

```
WITH
    MEMBER [Measures].[Store Profit] AS
        '([Measures].[Store Sales] - [Measures].[Store Cost])'
SELECT
    {[Measures].[Store Profit]} ON COLUMNS,
    {[Time].[Year].Members} ON ROWS
FROM [Sales]
WHERE
    [Store].[All Stores].[USA].[WA]
```



Time	Store Profit
+ 1997	28 977,57
+ 1998	

# Membres calculés

- On peut définir des membres calculés avec d'autres membres calculés
  - Exemple : membre calculé ProfitPercent (pourcentage du profit) :  
 $\text{ProfitPercent} = \text{Profit} / \text{Cost}$

WITH

```
MEMBER Measures.Profit AS 'Measures.Sales - Measures.Cost'
```

```
MEMBER Measures.ProfitPercent AS
```

```
'Measures.Profit / Measures.Cost', FORMAT_STRING = '#.##%
```

SELECT

```
{ Measures.Profit, Measures.ProfitPercent } ON COLUMNS
```

FROM Sales

- FORMAT\_STRING définit le format du membre calculé

# Membres calculés

- Exemple : pour comparer les chiffres de la compagnie entre 2012 et 2013, on peut définir un membre calculé sur le niveau 'Year', parallèle à 2012 et 2013, qui contiendra la différence entre eux :

```
WITH
    MEMBER Time.[12 to 13] AS 'Time.[2012] - Time.[2013]'
SELECT
    { Time.[12 to 13] } ON COLUMNS,
    Measures.MEMBERS ON ROWS
FROM Sales
```

# Membres calculés

- Exemple : Supposons qu'on veuille voir comment les profits ont évolués entre 2012 et 2013 :

WITH

MEMBER Measures.Profit AS 'Measures.Sales - Measures.Cost'

MEMBER Time.[12 to 13] AS 'Time.[2013] - Time.[2012]'

SELECT

{ Measures.Sales, Measures.Cost, Measures.Profit } ON COLUMNS,

{ Time.[2012], Time.[2013], Time.[12 to 13] } ON ROWS

FROM Sales

	<i>Sales</i>	<i>Cost</i>	<i>Profit</i>
<i>2012</i>	300	220	80
<i>2013</i>	350	210	140
<i>12 to 13</i>	50	-10	60

# Membre NULL

- La requête suivante calcule pour chaque mois la croissance des ventes comparée au mois précédent :

```
WITH  
    MEMBER Measures.[Sales Growth] AS  
        '(Sales) - (Sales, Time.PrevMember) '  
SELECT  
    {[Sales], [Sales Growth]} ON COLUMNS,  
    Month.MEMBERS ON ROWS  
FROM Sales
```

- La fonction PrevMember retourne le membre qui précède le membre actuel
- Pour le premier mois, il n'y a pas de mois précédent dans le cube
- Au lieu de retourner une erreur, MDX utilise la notion de membre NULL, représentant des membres qui n'existent pas

# Membre NULL

- La sémantique d'un membre NULL est :
  1. Quand la cellule contient un membre NULL, la valeur numérique de la cellule sera zéro, ainsi : `(Sales, [January].PrevMember) = 0`
  2. Toute fonction de membre appliquée à un membre NULL retourne NULL
  3. Quand un membre NULL est inclus dans un ensemble (Set), il est ignoré, ainsi :  
`{[September], [January].PrevMember} = {[September]}`

# Cellule EMPTY

- La notion cellule EMPTY est liée à celle de membre NULL
  - Quand une cellule est hors du cube alors sa valeur est NULL, et la valeur calculée pour cette cellule est 0 (zéro)
  - On peut avoir des cellules vides pas seulement hors du cube. C'est le cas où la données n'existe pas. La cellule est dite EMPTY
  - Lorsqu'une donnée n'existe pas, sa valeur numérique sera évaluée à 0

# Cellule EMPTY

- Pour traiter les cellules EMPTY, MDX propose :
  - Le prédicat `IsEmpty` qui peut être appliqué à un membre ou un tuple et retourner la valeur booléenne TRUE ou FALSE
  - La clause `NON EMPTY` qui permet de filtrer les résultats pour n'afficher que les valeurs des cellules NON EMPTY
    - La clause NON EMPTY ne peut être utilisée qu'au niveau des axes
  - La fonction `NONEMPTY()` qui permet d'exclure les cellules EMPTY lors de la sélection des membres d'un axe
    - La fonction NONEMPTY() peut être utilisée n'importe où dans la requête



# Cellule EMPTY

- Le prédicat IsEmpty :
  - Les villes où la mesure 'SalesAmount' n'est pas disponible en 2010, sont exclues du résultat, car leurs cellules sont vides

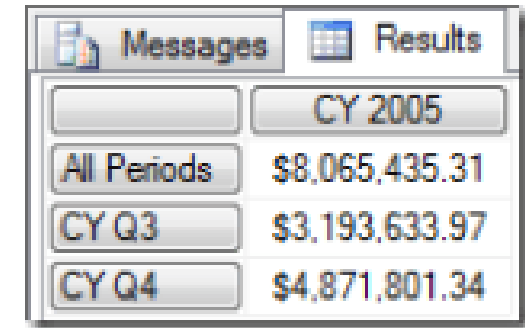
```
SELECT
    [City] ON 0,
    [Measures].[SalesAmount] ON 1
FROM
    [Sales]
WHERE
    NOT IsEmpty( ([Time].[2010], [Measures].[SalesAmount]) )
```

	New York	Paris	Barcelona	Madrid	Geneva	Lausanne	Zurich
Amount	768	4	2	1	128	56	64

# Cellule EMPTY

- La clause NON EMPTY :

```
SELECT
    ([Date].[Year].[2005]) ON 0,
    NON EMPTY [Date].[Quarter].MEMBERS ON 1
FROM [Sales]
WHERE [Measures].[SalesAmount]
```



The screenshot shows a 'Results' window with a pivot table. The columns are 'CY 2005' and 'SalesAmount'. The rows are 'All Periods', 'CY Q3', and 'CY Q4'. The values are \$8,065,435.31, \$3,193,633.97, and \$4,871,801.34 respectively. The 'Messages' tab is also visible.

	CY 2005
All Periods	\$8,065,435.31
CY Q3	\$3,193,633.97
CY Q4	\$4,871,801.34

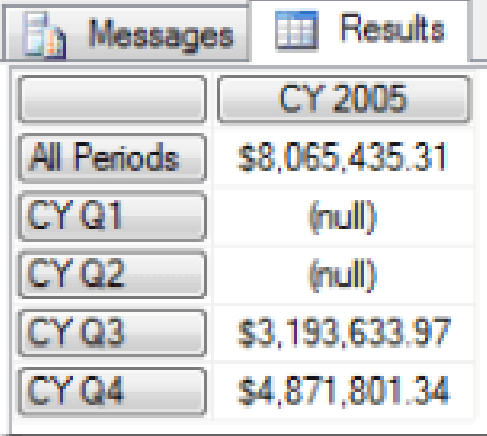
- Les trimestres Q1 et Q2 sont exclus du résultat car leurs cellules sont vides
- La clause NON EMPTY s'exécute comme suit : Après avoir décidé quels membres correspondent et quelles cellules vont être renvoyées, il faut éliminer les cellules vides

# Cellule EMPTY

- La fonction NONEMPTY() :

```
SELECT  
    ([Date].[Year].[2005]) ON 0,  
    NONEMPTY (  
        [Date].[Quarter].MEMBERS,  
        [Measures].[SalesAmount]) ON 1  
FROM [Sales]  
WHERE [Measures].[SalesAmount]
```

- Comme il existe des cellules pour d'autres années que 2005, la fonction Nonempty() n'élimine pas Q1 et Q2.
- La fonction Nonempty() n'est pas évaluée en dernière étape (comme NON EMPTY) mais lorsqu'on détermine les membres qui seront inclus dans l'axe.
- Ainsi, avant de savoir que la requête est limitée à 2005, Nonempty() a déjà déterminé quelles cellules seront exclues et incluses. Dans ce cas, aucune ligne n'est éliminée



	CY 2005
All Periods	\$8,065,435.31
CY Q1	(null)
CY Q2	(null)
CY Q3	\$3,193,633.97
CY Q4	\$4,871,801.34

# Partie 2 : Fonctions MDX

Points traités :

- Fonctions simples

- Fonctions de navigation sur les membres des dimensions

- Fonctions sur les ensembles (Sets)

# Fonctions simples

- `x.MEMBERS` :
  - Ensemble des membres d'un niveau ou d'une hiérarchie donnée
  - Inclus le membre ALL s'il existe
- `x.CHILDREN` :
  - Ensemble ordonnée des enfants du membre x

	Mesures
Region	• Actual
▣ All Regions	143 639 982,00
Central	37 893 162,00
Eastern	35 248 940,00
Southern	35 248 940,00
Western	35 248 940,00

	Mesures
Region	• Actual
Central	37 893 162,00
Eastern	35 248 940,00
Southern	35 248 940,00
Western	35 248 940,00

# Fonctions simples

- DESCENDANTS (x, l) :
  - Ensemble des descendants d'un membre x au niveau l

```
SELECT {[Measures].[Store Sales]} On COLUMNS,  
        DESCENDANTS ([Time].[1998], [Quarter]) ON ROWS  
FROM [SALES]
```

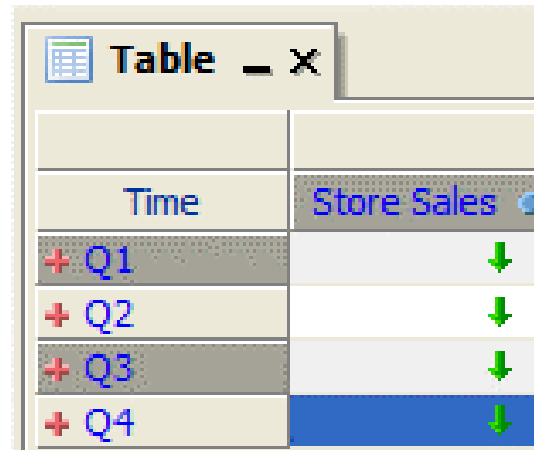


Table - X	
Time	Store Sales
+ Q1	↓
+ Q2	↓
+ Q3	↓
+ Q4	↓

# Fonctions simples

- **CrossJoin** : cette fonction retourne le produit croisé de deux ou plusieurs Sets

```
SELECT {  
    CrossJoin (  
        { ([Time].[1997].[Q1]), ([Time].[1997].[Q2]) },  
        { ([Measures].[Unit Sales]), ([Measures].[Store Sales]) }  
    ) } ON COLUMNS,  
    { ([Product].[Drink].Children) } ON ROWS  
FROM [Sales]
```

Table - X				
Product	Time			
	+ Q1		+ Q2	
	Unit Sales	Store Sales	Unit Sales	Store Sales
+ Alcoholic Beverages	↓ 312	↓ 635,00	↓ 289	↓ 636,57
+ Beverages	↓ 674	↓ 1 424,09	↓ 602	↓ 1 298,28
+ Dairy	↓ 193	↓ 317,29	↓ 201	↓ 330,69

# Fonctions de navigation entre les membres

- On a besoin de fonctions pour naviguer dans les hiérarchies de dimensions
  - Par exemple `[WA].Parent` (Etat de Washington) est équivalent à `[USA]`
  - Si l'on veut connaître les coordonnées d'une cellule dans le sous-espace de la requête on utilise `CurrentMember` :

`<dimension name>.CurrentMember`

`<level name>.CurrentMember`



# Fonctions de navigation entre les membres

- Exemple : on veut calculer le pourcentage des ventes dans chaque ville relativement à son état :

```
WITH
  MEMBER Measures.PercentageSales AS
    ' (Regions.CurrentMember, Sales) /
      (Regions.CurrentMember.Parent, Sales) ', FORMAT_STRING = '#.00%'
SELECT
  { Sales, PercentageSales } ON COLUMNS,
  Regions.Cities.MEMBERS ON ROWS
FROM Sales
```

# Fonctions de navigation entre les membres

Fonction	Signification	Remarque
Parent	Donne le parent du membre de dimension considéré	Déplacement vertical sur une hiérarchie, et on change ainsi de Niveau
FirstChild	Donne le premier fils du membre considéré	
LastChild	Donne le dernier fils du membre considéré	
NextMember	Donne le membre suivant du membre considéré	Déplacement horizontal à l'intérieur d'un même niveau
PrevMember	Donne le membre précédent fils du membre considéré	

- Exemples

- `[Aug].PrevMember` retourne `[Jul]` et les 2 membres appartiennent à `[Q3]`
- `[Oct].PrevMember` retourne `[Sep]`
  - Ici on traverse la hiérarchie en changeant de parents de `[Q4]` à `[Q3]`

# Fonctions de navigation entre les membres

- Exemple : On veut définir un nouveau membre calculé 'Sales Growth' montrant la croissance ou le déclin des ventes comparées avec le précédent mois, trimestre ou année :

```
WITH
    MEMBER Measures.[Sales Growth] AS
        '(Sales) - (Sales, Time.PrevMember) '
SELECT
    { [Sales], [Sales Growth] } ON COLUMNS,
    Month.MEMBERS ON ROWS      -- on peut utiliser Quarter ou Year
FROM Sales
```

# Fonctions de navigation entre les membres

- Exemple : Avec le même membre calculé, on veut observer l'évolution des ventes pour différents produits dans le dernier mois :

```
WITH  
    MEMBER Measures.[Sales Growth] AS  
        '(Sales) - (Sales, Time.PrevMember)'  
SELECT  
    { [Sales], [Sales Growth] } ON COLUMNS,  
    Product.MEMBERS ON ROWS  
FROM Sales
```

# Fonctions sur les sets

- Rappels :
  - Dans un « set », tous les éléments ont la même structure
  - Pour un set de membres : tous les membres doivent venir de la même dimension (même si ils appartiennent à différents niveaux) :
    - le set { [2012], [August] } est valide,
    - le set { [August], [Sales] } n'est pas valide.
  - Pour un set de tuples : la dimensionnalité doit être la même et les membres correspondants des tuples doivent venir de la même dimension :
    - le set { ([2012], [USA]), ([August], [WA]) } est valide,
    - le set { ([August], [WA]), ([USA], [2012]) } n'est pas valide, car l'ordre des dimensions dans le tuple est inversé

# Fonctions sur les sets

- Les fonctions peuvent avoir en entrée et en sortie des sets de membres ou de tuples
- Exemple de l'Union : `UNION( set1, set2 )` permet de combiner 2 ensembles ou plus
  - Exemple : on veut voir sur les axes résultats : l'Europe, les USA, tous les états des USA, toutes les villes dans l'état WA, et l'Asie (scénario de Drilldown typique)
    - On définit alors un set :  
`{ [Europe], [USA], [USA].Children, [WA].Children, [Asia] }`
    - L'équivalent avec l'opérateur UNION sera :  
`UNION( { [Europe], [USA] }, UNION( [USA.Children], UNION( [WA].Children, { [Asia] } ) ) )`
  - Certaines implémentations de MDX disposent de l'opérateur + (plus) pour faire l'union de 2 sets : `set1 + set2 + ...`

# Fonctions sur les sets

Fonction	Syntaxe	Description
Head	Head(<< Set >> [, << Numeric Expression >>])	Éléments de tête d'un set
Tail	Tail(<< Set >> [, << Numeric Expression >>])	Derniers éléments d'un Set
Subset	Subset(<< Set >>, << Start >> [, << Count >>])	Sous-ensemble d'éléments d'un set
TopCount	TopCount(<< Set >>, << Count >> [, << Numeric Expression >>])	Les premiers éléments ayant la plus grande valeur de la mesure
Order	Order (<< Set <<, {<<String Expression>>   <<Numeric Expression >>} [, ASC   DESC   BASC   BDESC])	Tri des éléments d'un set
Filter	Filter(<< Set >>, << conditions >>)	Les éléments d'un set qui satisfont le filtre

# Fonctions sur les sets

- Exemple d'utilisation des fonctions Topcount, Tail et Head :

```
SELECT
    { ([Measures].[Unit Sales]) } ON COLUMNS,
    { (Head([Time].Children, 2)),
      (Tail([Time].Children, 3)),
      (Topcount([Time].Children, 1, [Measures].[Unit Sales])) } ON ROWS
FROM [Sales]
```

- Head : retourne les 2 premiers « Children » de Time
- Tail : retourne les 3 derniers « Children » de Time
- Topcount : retourne le (1) « Children » de Time ayant la meilleure « Unit Sales »



# Fonctions sur les sets

- Exemple avec Filter : la requête suivante n'affiche que les années où la mesure [Unit Sales] est supérieur à 70000 :

```
SELECT
    { ([Measures].[Unit Sales]) } ON COLUMNS,
    { Filter ([Time].Children, ([Measures].[Unit Sales]) > 70000)
    } ON ROWS
FROM [Sales]
```

# Fonctions sur les sets

- Exemple qui renvoie les cinq sous-catégories de produits les plus vendus, quelle que soit la hiérarchie , selon « Unit Sales »
  - Head est utilisée pour renvoyer uniquement les 5 premiers Sets dans le résultat après que le résultat soit ordonné avec Order :

```
SELECT
    [Measures].[Unit Sales] ON 0,
    Head (Order ([Product].[Categories].[SubCategory].MEMBERS,
                [Measures].[Unit Sales], BDESC) , 5 ) ON 1
FROM [Sales]
```

- La fonction Order peut être hiérarchique (si on utilise ASC ou DESC) ou non hiérarchique (en utilisant BASC ou BDESC)
  - B signifie « Break hierarchy »

# Partie 3 : Expressions avancées en MDX

Points traités :

- Analyse comparative (ParallelPeriod)

- Calcul cumulatif

- Expressions conditionnelles (IIF)

# Analyse comparative

- L'analyse comparative consiste à présenter à côté d'une mesure, la même mesure sur une période parallèle, qui lui est antérieure

- On utilise la fonction ParallelPeriod :

```
ParallelPeriod(["Level" [, "expression numérique" [, "MEMBERS" ]]])
```

- Où 'Level' représente le niveau hiérarchique sur lequel on veut remonter dans le temps, l'expression numérique donne le nombre de périodes, et membre permet de fixer un membre sur la dimension temps

# Analyse comparative

- Exemple : on veut afficher la mesure [Unit Sales] en parallèle sur le trimestre en cours et sur le trimestre antérieur :

```
WITH
    MEMBER [Measures].[Unit Sales Q-1] AS
        ( ParallelPeriod( [Time].[2012].[Q1], 1) )
SELECT
    { ([Measures].[Unit Sales]), ([Measures].[Unit Sales Q-1]) }
    ON COLUMNS,
    { ([Time].Children) } ON ROWS
FROM [Sales]
```

# Calcul cumulatif

- Ce calcul permet de calculer des cumuls sur une période de temps
- Exemple : calcul du cumul de la mesure [Unit Sales] durant l'année 2012
  - Dans un premier temps, on doit retrouver tous les mois de l'année 2012
  - On peut envisager un accès par membre, mais l'expression correspondante pourrait être longue et dépendante des trimestres :

```
SELECT
    { ([Measures].[Unit Sales]) } ON COLUMNS,
    { ([Time].[2012].[Q1].Children),
      ([Time].[2012].[Q2].Children),
      ([Time].[2012].[Q3].Children),
      ([Time].[2012].[Q4].Children) } ON ROWS
FROM [Sales]
```

# Calcul cumulatif

- L'utilisation de la fonction Descendants est plus intéressant :
  - 1<sup>er</sup> paramètre : retourne un ensemble de descendants d'un membre sur un niveau ou d'une distance spécifiée
  - 2<sup>ème</sup> paramètre peut être soit une expression de niveau spécifique, soit un nombre indiquant le nombre de niveaux à parcourir
- Exemple : Calcul des descendants d'une distance égale à 2 niveaux (mois) à partir de l'année 2012 :

```
SELECT
    { ([Measures].[Unit Sales]) } ON COLUMNS,
    { Descendants([Time].[2012],2) } ON ROWS
FROM [Sales]
```

# Calcul cumulatif

- Exemple : calcul des descendants du niveau mois du membre, dont la valeur est le 2<sup>ème</sup> semestre :

```
SELECT
    { ([Measures].[Unit Sales]) } ON COLUMNS,
    { Descendants([Time].[2012].[Q2],[Time].[Month]) } ON ROWS
FROM [Sales]
```



# Expressions conditionnelle (IIF)

- IIF permet de réaliser des tests conditionnels
- Dans l'exemple qui suit, on construit plusieurs membres dont :
  - `[Measures].[Q-1]` : donne [Unit Sales] au trimestre précédent
  - `[Measures].[Evolution]` : donne l'évolution de [Unit Sales] entre deux trimestres consécutifs
  - `[Measures].[%]` : donne l'évolution en pourcentage
  - `[Measures].[Performance]` : renvoie une chaîne de caractères qui nous permet d'appliquer des règles de gestion liées à des conditions

# Expressions conditionnelle (IIF)

```
WITH
  MEMBER [Measures].[Q-1] AS
    ( ParallelPeriod([Time].[2012].[Q1],1, [Time].CurrentMember),
      [Measures].[Unit Sales] )
  MEMBER [Measures].[Evolution] AS
    ( ([Time].CurrentMember, [Measures].[Unit Sales]) -
      (ParallelPeriod([Time].[2012].[Q1],1, [Time].CurrentMember),
        [Measures].[Unit Sales]) )
  MEMBER [Measures].[%] AS
    ( [Measures].[Evolution] /
      (ParallelPeriod([Time].[2012].[Q1],1, [Time].CurrentMember),
        [Measures].[Unit Sales]) ), format_string = "Percent"
```

# Expressions conditionnelle (IIF) (suite)

```
MEMBER [Measures].[Performance] AS
    IIF([Measures].[Q-1]<>0,
        IIF([Measures].[Q] < 0, "-",
            IIF([Measures].[Q] > 0 and [Measures].[Q] < 0.01, "+",
                IIF([Measures].[Q] > 0.01 and [Measures].[Q] < 0.05, "++",
                    "better performance") -- Mieux que 5%
            )
        ),
        "null")
SELECT
    { ([Measures].[Unit Sales]), ([Measures].[Q-1]),
      ([Measures].[Evolution]), ([Measures].[Q]) ,
      ([Measures].[Performance]) } ON COLUMNS,
      Descendants([Time], 1) ON ROWS
FROM [Sales]
```