A. Introduction

L'intégralité de ce chapitre est consacré à l'idée de « visuel augmenté », à savoir l'imbrication des visuels et des formules DAX en vue d'obtenir une meilleure représentation de l'information : dynamique (une partie du résultat est liée aux choix de l'utilisateur), conditionnelle (certains points de données sont mis en avant) ou encore contextualisée (l'information est mise dans un contexte plus vaste, le temps par exemple).

C'est pourquoi nous commencerons par un rappel des notions qu'il est essentiel de maîtriser. Nous ne rentrerons cependant pas dans le détail. Pour cela vous pouvez consulter notre ouvrage *Power BI Desktop, Reporting et analyse de données au quotidien (3e édition)*.

B. Fondamentaux

1. Les mesures et les colonnes

Au-delà de l'aspect lié au stockage des données qui est optimisé dans le cas des mesures, la différence essentielle réside dans le fait que les mesures sont dynamiques, alors que les colonnes sont fixes.

Les colonnes sont en effet calculées lors de leur création initiale, puis à chaque actualisation du rapport. Un segment n'a donc aucun effet sur la valeur stockée dans une colonne. C'est exactement l'inverse pour les mesures : elles ne sont calculées qu'au moment où elles sont déposées dans un rapport, et sont donc impactées par un segment.

Prenez l'exemple du classement de produits : la fonction RANKX permet de classer en fonction d'une valeur (par exemple le montant) et peut être utilisée pour créer une colonne ou une mesure. La colonne va classer les produits en fonction de la quantité, mais de la quantité globale (c'est-à-dire, toutes années, toutes catégories de produits, tous types de clientèle). Et ce classement sera figé, et ne changera qu'au prochain chargement de données.

Calculé sous forme de mesure, le classement devient sensible aux segments présents sur le rapport : l'utilisateur pourra donc obtenir un classement en fonction d'une année, de l'année en cours, d'une catégorie de produits, ou encore d'un type de clientèle. A chaque changement, la mesure classement est recalculée.

L'inconvénient, c'est que l'écriture de la mesure est plus complexe que celui de la colonne.

2. Le contexte de filtre, le contexte de ligne

Il ne s'agit pas pour nous ici de reprendre intégralement ces notions, mais simplement de rappeler que le contexte de ligne est invoqué automatiquement lors de la création d'une colonne ainsi que lors de l'utilisation d'une fonction de type itérateur (en particulier les fonctions X – SUMX, etc.).

Le contexte de filtre, bien plus courant, est défini par les visuels (tables, graphiques, segments) présents sur le rapport (et parfois sur les autres rapports). Mais il peut aussi être manipulé par formule, et c'est précisément le rôle de la fonction CALCULATE.

Le contexte de ligne filtre la table sur laquelle il s'applique, et n'en retient qu'une ligne. L'utilisation des fonctions RELATED ou RELATEDTABLE permet cependant au contexte de ligne de se propager vers d'autres tables, dans le sens N vers 1 de la relation pour la première, dans le sens 1 à N pour la seconde.

Le contexte de filtre filtre le modèle dans son ensemble, selon le sens de propagation de 1 à N le long des relations. L'utilisation de la fonction CROSSFILTER, ou d'une relation bidirectionnelle, permet toutefois au filtre de se propager dans le sens N à 1.

Enfin, dans certains cas, le contexte de ligne est transformé en contexte de filtre, selon le principe de la transition de contexte que nous avons déjà bien abordé.

3. Aide-mémoire DAX et mise en forme

Afin d'assurer la lisibilité des formules, et notamment lorsqu'elles deviennent complexes, certaines règles de mises en forme peuvent être suivies :

- Les colonnes référencées le sont toujours en précisant le nom de la table d'abord, alors que les mesures sont simplement référencées entre crochets : cela aide en particulier à mieux « voir » le CALCULATE implicite.
- Ajoutez autant de commentaires que nécessaire, précédés d'un //.
- Facilitez la vue des parenthèses ouvrantes et fermantes.
- Allez à la ligne autant de fois que vous le voulez.

```
[Montant moyen facturé] =
   AVERAGEX (
       Date[Date],
       [Montant facturé]
   )
```

Notez aussi qu'une variante peut consister à décaler la virgule sur la ligne suivante, pour faciliter la mise en commentaire d'une ligne et par conséquent les tests :

```
[Mesure] =
   CALCULATE (
       expression
       , filtre1
       , filtre2
       , filtre3
   )
```

Écrite comme ça, la formule permet facilement de mettre une partie du code en commentaire (ici, le filtre 3) :

```
[Mesure] =
   CALCULATE (
       expression
       , filtre1
       , filtre2
//     ; filtre3
   )
```

Voici un tableau des formules clés du DAX, base sur laquelle nous allons pouvoir bâtir des formules plus complexes dans le reste de ce chapitre.

Attention, le modèle est un peu différent de celui que nous utilisons, puisque les tables **Commandes** et **Détails de commandes** d'une part, **Livres** et **Catégories** d'autre part sont ici restées distinctes. C'est un modèle entête-détail en flocon.

SUM	Pour faire la somme des valeurs dans une colonne, une fois le contexte de filtre appliqué
`SUM(Colonne)`	`quantité = SUM('Detail des commandes'[Quantité])`
SUMX	Pour effectuer un calcul sur chaque ligne de la table passée en paramètre
`SUMX(Table, Expression)`	`montant = SUMX( 'Detail des commandes', [quantité] * 'Detail des commandes'[Prix de vente] )`

RELATED	Argument de SUMX notamment, permet de voir les colonnes des tables situées « au-dessus » de celle qui est passée en paramètre

	`montant prix catalogue =`
	`SUMX(`
	`  'Detail des commandes',`
	`  [quantité] * RELATED('Livres'[Prix catalogue])`
	`)`

AVERAGEX	Pour calculer une moyenne au niveau de détail que vous souhaitez (VALUES)

`AVERAGEX(Table, Expression)`
Notez que VALUES retourne une table
Ce schéma fonctionne également avec MINX et MAXX
	`montant mensuel moyen =`
	`AVERAGEX(`
	`  VALUES(Datum[AAMM]),`
	`  [montant]`
	`)`

Compter avec COUNTROWS et DISTINCTCOUNT	
`COUNTROWS(Table)`	`cpt client = COUNTROWS(Clients)`
`DISTINCTCOUNT (Colonne)`	`nb de clients ayant passé des commandes =` `DISTINCTCOUNT(Commandes[Code client])`

	`% de clients ayant commandé / total client =`
	`DISTINCTCOUNT(Commandes[Code client]) /`
	`COUNTROWS(Clients)`

`FILTER(Table, Expression de filtre)`	`cpt clients nantais =`
	`COUNTROWS(`
	`  FILTER(`
	`  Clients,`
	`  Clients[Ville] = "Nantes"`
	`  )`
	`)`

	Un peu plus complexe…
	clients ayant passé plus de 2 commandes = COUNTROWS(FILTER(SUMMARIZE(Commandes, Commandes[Code client], "nb comm", COUNT(Commandes[Code client])), [nb comm] > 1))
	clients n'ayant pas commandé = COUNTROWS(EXCEPT(VALUES(Clients[Code client]), VALUES(Commandes[Code client])))
CALCULATE	Pour modifier un filtre (ajouter, supprimer, remplacer) CALCULATE est la fonction majeure du DAX
CALCULATE(expression, expression de filtre)	quantité Hachette = CALCULATE([quantité], Vendeurs[Nom vendeur] = "Hachette")
	Vendeurs[Nom vendeur] IN { "Hachette" , "Favreau" }
	quantité Hachette à Nantes = CALCULATE([quantité], Vendeurs[Nom vendeur] = "Hachette", Clients[Ville] = "Nantes")
	quantité annuelle = CALCULATE([quantité], ALL(Calendrier[Annee Mois]))

	`quantité tous livres =`
	`CALCULATE(`
	`  [quantité],`
	`//  ALL(Livres)`
	`//  ALL()`
	`  ALLSELECTED()`
	`)`

USERELATIONSHIP	Argument de CALCULATE, permet d'activer le temps d'un calcul une relation (physique) inactive

	`quantité livraison =`
	`CALCULATE(`
	`  [quantité],`
	`  USERELATIONSHIP(`
	`  Datum[Date],`
	`  'Detail des commandes'[Date de livraison]`
	`  )`
	`)`

CROSSFILTER	Argument de CALCULATE, permet d'inverser le temps d'un calcul le sens d'une relation

	`quantité livraison =`
	`CALCULATE(`
	`  [quantité],`
	`  CROSSFILTER (`
	`  Datum[Date],`
	`  'Detail des commandes'[Date de livraison],`
	`  Both`
	`  )`
	`)`

	`cpt catégories =`
	`CALCULATE(`
	`  DISTINCTCOUNT(Livres[Code catégorie]),`
	`  CROSSFILTER(`
	`  'Detail des commandes'[Numéro livre],`
	`  Livres[Numéro livre],`
	`  Both`
	`  )`
	`)`

IF ISINSCOPE	Faire un calcul seulement si.

	`% catégorie / suspense =`
	`IF(`
	`  ISINSCOPE(Livres[Signification]),`
	`  [quantité] / [quantité Suspense] -1,`
	`  "n/a"`
	`)`

Time Intelligence	Comparer deux périodes

C'est tout ce qu'il y a besoin de connaître !
Le dernier argument peut être YEAR, QUARTER, MONTH ou DAY
	`quantité M-3 (comparaison) =`
	`CALCULATE(`
	`  [quantité],`
	`  DATEADD(`
	`  Datum[Date],`
	`  -3, MONTH`
	`  )`
	`)`

Time Intelligence	Cumuler des résultats
Variantes : TOTALQTD, TOTALMTD	`TOTALYTD([quantité], Datum[Date])`
SUMMARIZECOLUMNS	Pour créer une table physique (très utile pour créer des tables d'agrégats)

	`agrégat ville année =`
	`SUMMARIZECOLUMNS(`
	`  Clients[Ville],`
	`  Datum[Année],`
	`  FILTER(`
	`  Clients,`
	`  Clients[Ville] in {"ANGERS", "NANTES"}`
	`  ),`
	`  "qte agr.", [quantité],`
	`  "montant agr.", [montant])`

4. Utilisation de la vue de requête DAX

La vue de requête DAX est particulièrement utile dans quatre situations :

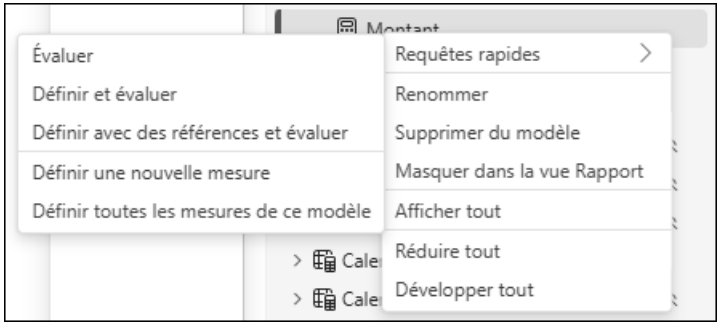
- prendre en main un rapport et en comprendre l'organisation des mesures ;
- prévisualiser le résultat d'une formule grâce à la fonction **EVALUATE** ou les 100 premières lignes d'une table ;
- faire des mises à jour groupées ;
- utiliser Copilot pour écrire des formules DAX.

a. Prendre en main un rapport

Vous prenez le relais sur le développement ou la maintenance d'un rapport, et vous avez besoin de faire un tour d'horizon des mesures qu'il contient. Jusque-là, il fallait pour cela recourir à des outils externes.

Dans la **Vue de requête DAX**, faites un clic droit sur une mesure et choisissez **Requêtes rapides** puis une des options suivantes :

- **évaluer** : pour calculer la mesure ;
- **définir et évaluer** : comme ci-dessus, avec l'affichage de la formule et la possibilité de la modifier et de mettre à jour le modèle ;
- **définir avec des références et évaluer** : définit la mesure et les mesures auxquelles elle fait appel (afficher les formules), et calcule le résultat ;
- **définir toutes les mesures de ce modèle** : pour afficher toutes les formules utilisées dans le rapport.



Sur l'exemple suivant, nous avons défini toutes les mesures du modèle, résultant en presque 250 lignes de DAX :

- Passez la souris sur une mesure pour afficher une fenêtre pop-up avec le code de la mesure citée.

Notez au passage le Code Lens : c'est l'indication en gris **Mettre à jour le modèle** : remplacer la mesure. Comme son intitulé l'indique, le Code Lens permet d'implémenter directement les modifications faites sur une mesure à partir de la **Vue de requête DAX**.

Le bouton **Mettre à jour le modèle avec les modifications (26)** en haut de page permet d'implémenter toutes les modifications et de mettre à jour toutes les mesures concernées.

b. Faire des mises à jour groupées

Que vous affichiez la formule d'une seule mesure, ou de toutes les mesures, la **Vue de requête DAX** offre un outil classique de recherche/remplacement, bien utile lorsque vous avez besoin d'effectuer une mise à jour groupée.

- Dans l'onglet **Accueil** cliquez sur le bouton **Remplacer** du groupe **Modification** ou utilisez le raccourci Ctrl H).