A. Introduction

SI vous souhaitez approfondir vos connaissances sur la fonction CALCULATE, je vous conseille mon précédent ouvrage, Power BI Desktop : Renforcer, approfondir, explorer, aux Éditions ENI, dont le chapitre 3 décrit notamment des motifs courants et des analyses fréquentes rendues possibles avec CALCULATE. Je ne reviendrai pas sur ces points, mais je résume les caractéristiques essentielles de cette fonction et j'ajoute ici d'autres aspects de son fonctionnement.

B. Les principes de CALCULATE

Vous l'avez vu depuis le début de cet ouvrage, la fonction CALCULATE revient tout le temps. La raison en est simple, et fondamentale : CALCULATE a le pouvoir de modifier le contexte de filtre généré par le visuel, les segments, le rapport en général.

1. La syntaxe

Elle est très simple :

```
CALCULATE ( <Expression> [ , <Filtre1> ] [ , <Filtre2> ] [ , ...])
```

Mais ceci amène à de nombreuses remarques.

CALCULATE et la transition de contexte

Tout d'abord, CALCULATE peut s'utiliser avec pour seul argument l'expression qu'il s'agit de calculer. En effet, c'est un rôle second de la fonction : lorsqu'elle est invoquée dans un contexte de ligne (c'est-à-dire lors de la création d'une colonne ou lors de son utilisation dans une fonction itérative – SUMX par exemple), la formule `CALCULATE(<expression>)` déclenche une **transition de contexte**, et la ligne dans son intégralité devient un filtre, qui se propage donc aux tables du modèle.

Ce phénomène est loin d'être rare : en fait, il est essentiel de se rappeler qu'à chaque fois qu'une mesure est appelée dans une formule, un CALCULATE implicite est systématiquement ajouté.

Prenons un exemple. La mesure `[Montant facturé]` est définie par la formule :

```
[Montant facturé] = SUM(Ventes[Montant])
```

Si l'on souhaite connaître le montant moyen facturé chaque jour, la formule s'écrit :

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

Soit, en développant la formule en incluant le CALCULATE implicite :

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

CALCULATE étant invoqué dans le cadre d'un contexte de ligne (dû à la fonction AVERAGEX, itérateur), la transition de contexte est déclenchée, et la date est donc propagée à la table `Ventes`. Celle-ci est donc filtrée, il ne reste que les lignes de la table `Ventes` correspondant à la date du contexte de ligne. C'est ce qui permet le calcul de la moyenne, les montants restants sont additionnés, puis le résultat est divisé par le nombre de lignes.

Exercice : pour mettre en pratique, vous pouvez réaliser l'exercice Combien de livres par catégories ? à la fin de ce chapitre.

Un nombre de filtres non limité

La seconde remarque tient au nombre de filtres de CALCULATE : vous pouvez en ajouter autant que vous le souhaitez. Les filtres sont tous traités simultanément (l'ordre des filtres n'a donc aucune importance) – mais nous reviendrons sur cette notion, lorsque nous distinguerons les filtres proprement dits et les modificateurs.

Un filtre dans CALCULATE peut être une table (c'est-à-dire le résultat d'une fonction de table, autrement dit une liste de valeurs), ou un prédicat booléen (Vrai/Faux).

Voici un exemple du premier cas, retournant une liste de valeurs :

```
calculate FILTER =
CALCULATE (
      [montant] ,
      FILTER (
            produits ,
            produits[couleur] = "Rouge"
      )
)
```

Voici un exemple du second, retournant vrai ou faux :

```
CALCULATE (
      [Montant] ,
      'Produit'[Couleur] = 'Rouge'
)
```

Il est utile et important de noter que cette dernière syntaxe est un raccourci, simple à écrire, qui est systématiquement développé par DAX en :

```
CALCULATE (
      [Montant] ,
      FILTER (
           ALL(Produit),
           'Produit'[Couleur] = 'Rouge'
      )
)
```

Le filtre booléen Vrai/Faux cache la fonction de table FILTER, dont le résultat est une liste de valeurs.

Attention, la première et la deuxième expression ne produisent pas toujours le même résultat. Un exercice vous permettra de creuser ce point.

2. Comment CALCULATE modifie le contexte de filtre

Je vous renvoie et je vous encourage à relire la section consacrée à ce point dans le chapitre Les principes fondamentaux du DAX du présent ouvrage (La fonction CALCULATE - Les trois façons de modifier le contexte de filtre (remplacer, ajouter, supprimer)). Il est essentiel que vous compreniez bien ces notions.

En résumé, CALCULATE agit de trois manières :

- Remplacer le contexte de filtre : lorsque le filtre explicite donné comme argument de la fonction CALCULATE porte sur la même colonne que celle qui définit le contexte implicite (celui que génère le rapport), CALCULATE remplace ce dernier par le premier.
- Ajouter au contexte de filtre : lorsque le filtre explicite porte sur une nouvelle colonne, le nouveau filtre est ajouté au filtre implicite.

- Supprimer le contexte de filtre : c'est le rôle de la fonction ALL et de ses variantes.

Exercice : pour mettre en pratique, vous pouvez réaliser l'exercice Les arguments de filtres complexes à la fin de ce chapitre.

C. Les arguments de filtres complexes (AND, OR)

Il existe de nombreux cas de figure : vous en trouverez à la fin de cette section un tableau récapitulatif simplifié.

1. Sur une colonne

Par filtre complexe, il faut comprendre un filtre portant plusieurs fois sur la même colonne, pour encadrer des dates par exemple ou encore, indiquer plusieurs catégories.

Deux cas se présentent : celui où les deux conditions doivent s'appliquer simultanément et celui où l'une ou l'autre des conditions doit s'appliquer.

Lorsque la date doit être comprise entre deux bornes, alors les conditions doivent s'appliquer simultanément. C'est ce qui se passe lorsque vous indiquez plusieurs filtres dans CALCULATE :

```
montant janv-mars 2019 =
CALCULATE(
    [montant] ,
    Datum[Date] >= DATE(2019,01,01) ,
    Datum[Date] <= DATE(2019,03,31)
)
```

Annee mois	montant	montant janv-mars 2019
201901	€ 8 621,00	24 243,00 €
201902	€ 8 278,00	24 243,00 €
201903	€ 7 344,00	24 243,00 €
201904	€ 5 420,00	24 243,00 €
201905	€ 5 539,00	24 243,00 €
201906	€ 7 345,00	24 243,00 €
201907	€ 7 329,00	24 243,00 €
201908	€ 5 535,00	24 243,00 €
201909	€ 5 823,00	24 243,00 €
201910	€ 5 039,00	24 243,00 €
201911	€ 5 315,00	24 243,00 €
201912		24 243,00 €
Total	**€ 71 588,00**	**24 243,00 €**

Il n'est sans doute pas inutile de rappeler que la table **Datum** *ayant été marquée comme table de dates, le contexte de filtre induit par le visuel (donc ici le champ* **Annee mois***) est ramené à un filtre sur un ensemble de dates (donc sur le champ* **Date***). Dans le CALCULATE ci-dessus, le filtre porte lui aussi sur le champ* **Date***, il remplace donc le contexte de filtre existant. C'est la raison pour laquelle nous retrouvons le même montant sur toutes les lignes du tableau. Ceci explique également pourquoi la ligne 201912 apparaît : bien qu'il n'y ait pas de montant ce mois-là, puisqu'il y en a un pour janv-mars, cette ligne est rajoutée au tableau.*

Par défaut, les deux conditions dans CALCULATE sont liées par un ET (AND). La formule précédente pourrait aussi s'écrire à l'aide de l'opérateur && :

```
montant janv-mars 2019 =
CALCULATE(
    [montant] ,
    Datum[Date] >= DATE(2019,01,01)
    && Datum[Date] <= DATE(2019,03,31)
)
```

Elle peut aussi s'écrire avec l'opérateur AND :

```
montant janv-mars 2019 =
CALCULATE(
      [montant] ,
      AND(
            Datum[Date] >= DATE(2019,01,01),
            Datum[Date] <= DATE(2019,03,31)
      )
)
```

Attention, l'opérateur AND (de même que OR) n'accepte que deux arguments. Si votre filtre doit porter sur trois conditions ou plus, alors privilégiez &&.

Le cas inverse est celui où l'une ou l'autre des conditions doit s'appliquer. Dans ce cas, c'est l'opérateur OU (OR) qui doit lier les conditions (symbole || , appelé *double pipe*, c'est-à-dire les touches AltGr 6) :

```
montant produits verts ou turquoises =
CALCULATE(
    [montant] ,
    produits[couleur] = "Vert"
    || produits[couleur] = "Turquoise"
)
```

La formule précédente pourrait aussi s'écrire à l'aide de l'opérateur IN :

```
montant produits verts ou turquoises =
CALCULATE(
    [montant] ,
    produits[couleur] IN { "Vert" , "Turquoise" }
)
```

Notez bien les accolades pour énumérer la liste des catégories.

Elle peut aussi s'écrire avec l'opérateur OR :

```
montant produits verts ou turquoises =
CALCULATE(
      [montant] ,
      OR(
            produits[couleur] = "Vert",
            produits[couleur] = "Turquoise"
      )
)
```

Attention, l'opérateur OR n'accepte que deux arguments. Si votre filtre doit porter sur trois conditions ou plus, alors privilégiez ||.

2. Sur plusieurs colonnes

Lorsqu'il s'agit de faire porter les filtres sur plusieurs colonnes, de la même table ou de tables différentes, il faut à nouveau distinguer le cas où ils s'appliquent simultanément ou si l'un ou l'autre doit s'appliquer.

Dans le premier cas, l'écriture reste simple :

```
montant produits verts en 2019 =
CALCULATE(
    [montant] ,
    produits[couleur] = "Vert" ,
    Datum[Annee] = "2019"
)
```

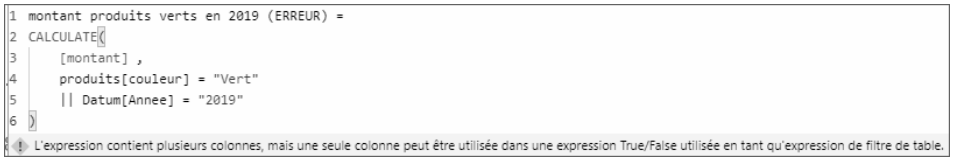
201810	€ 5 928,00	
201811	€ 6 338,00	
201812	€ 4 270,00	
201901	€ 8 621,00	924,00 €
201902	€ 8 278,00	386,00 €
201903	€ 7 344,00	330,00 €
201904	€ 5 420,00	258,00 €

En revanche, l'utilisation de && n'est plus possible. La formule ci-dessous génère une erreur :

```
montant produits verts en 2019 (ERREUR) =
CALCULATE(
    [montant] ,
    produits[couleur] = "Vert"
    && Datum[Annee] = "2019"
)
```

Lorsque l'une ou l'autre condition doit être appliquée (opérateur OU), la syntaxe simple n'est pas non plus possible, et génère la même erreur que précédemment :

```
montant produits verts ou 2019 (ERREUR) =
CALCULATE(
    [montant] ,
    produits[couleur] = "Vert"
    || Datum[Annee] = "2019"
)
```



Dans ce cas, il est nécessaire de recourir à la syntaxe développée, complexe, notamment si les deux colonnes ne proviennent pas de la même table :

```
montant produits verts ou 2019 =
CALCULATE(
    [montant] ,
    FILTER(
        CROSSJOIN(
            ALL(produits[couleur]) ,
            ALL(Datum[Annee])
        ) ,
        produits[couleur] = "Vert"
        || Datum[Annee] = "2019"
    )
)
```

Ici, la fonction CROSSJOIN retourne une table avec toutes les combinaisons possibles des deux colonnes.