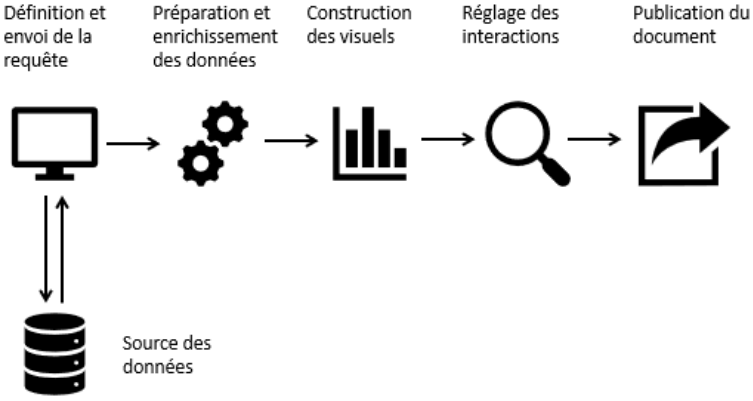
A. Les concepts clés

1. Circulation des données

Comprendre la façon dont circulent et sont stockées les données lorsque vous utilisez Power BI Desktop peut s'avérer utile, notamment lors du choix de la connexion, lors de l'actualisation, de la création des colonnes ou des mesures, et de l'utilisation même du rapport.

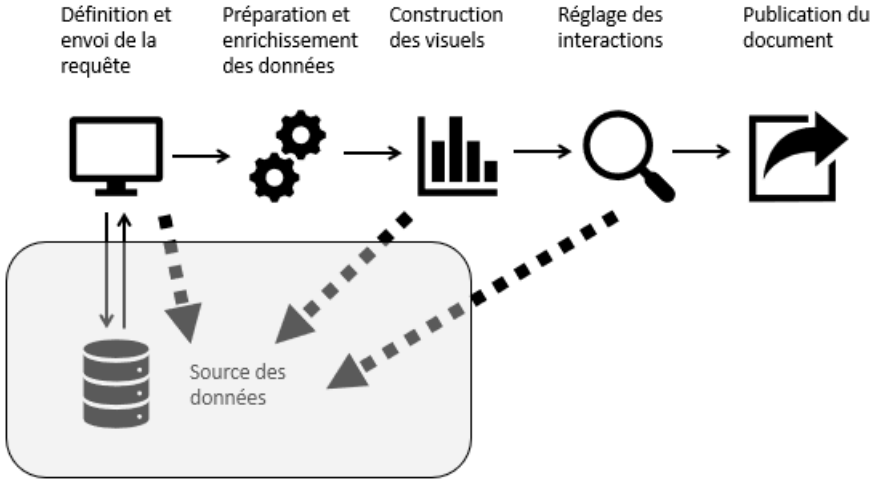
Reprenons le schéma du chapitre précédent :



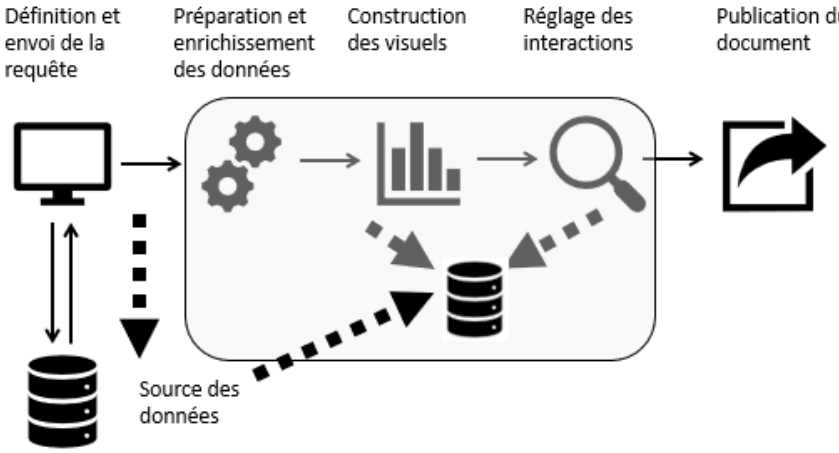
Le premier point clé concerne le type de connexion : la volumétrie des données, ou le besoin de temps (quasi) réel, peuvent vous amener à utiliser une connexion en direct sur votre base de données (mode Direct) – dans ce cas en effet, les données restent dans la base, qui est interrogée à chaque manipulation dans le fichier au prix d'un effort important sur la base.

Les types de connexion sont précisés dans la section suivante.

Dans ce premier cas, les données circulent « à la demande » et ne sont pas stockées localement. Une opération aussi simple qu'afficher un total au bas d'un tableau ou filtrer un graphique amène à réinterroger la source. Les performances dépendent des performances du système de base de données :



Dans les autres cas, l'application sera plus performante si les données sont rapatriées localement (mode Import) : la source de données n'est interrogée qu'une fois, lors de l'exécution ou de l'actualisation de la requête, puis les données sont stockées localement, à l'intérieur du fichier.



Dans ce cadre, Power BI repose sur un outil très performant : le moteur de stockage tabulaire, VertiPaq, accélère considérablement la mise en place des visuels par la sélection des champs, les interrogations du modèle à chaque utilisation et à chaque calcul d'une mesure, la mise à jour des visuels lors de l'utilisation des filtres et des interactions en général.

Il est également possible de mettre en place des modèles composites, dont une partie des tables (ou des sources), sont exploitées en mode Import, et les autres en mode Direct. Par exemple, des tables Produits ou Sites géographiques, qui évoluent peu, peuvent être importées, mais les tables des transactions, ou des chaînes de production, qui ont parfois besoin d'être intérrogées en quasi-temps réel, peuvent rester en mode Direct.

2. Actualisation

Dans le cadre d'une connexion avec importation des données, l'actualisation dans Power BI Desktop se fait exclusivement à l'aide de l'outil **Actualiser**.

Dans le cadre d'une connexion directe à la source de données, nous l'avons vu, l'actualisation dans Power BI Desktop se fait à chaque action sur un visuel. Elle peut aussi être déclenchée manuellement à l'aide de l'outil **Actualiser** dans l'onglet **Accueil** :

Enfin, c'est dans le cadre de la publication sur Power BI Service (le serveur Power BI) que les possibilités d'actualisation sont les plus complètes.

Lorsque vous publiez un rapport sur Power BI Service, un jeu de données est également envoyé sur le serveur, et disponible aux côtés des rapports et des tableaux de bord (composés par l'utilisateur à partir des visuels présents dans les rapports).

Ce jeu de données peut faire l'objet d'une actualisation immédiate (manuelle), automatique (notamment si le fichier source est stocké dans SharePoint ou OneDrive) ou programmée (planifiée).

Le plus souvent, il faut avoir mis en place une passerelle entre le serveur Power BI et le serveur de base de données, pour que la mise à jour puisse se faire.

Si l'utilisateur a créé un tableau de bord, en épinglant différents visuels disponibles dans les rapports auxquels il a accès, une fonctionnalité **Actualiser les vignettes du tableau de bord** est disponible, pour forcer l'actualisation (en dehors de cette actualisation manuelle, une actualisation automatique a lieu toutes les 15 ou 30 minutes).

Pour certains types de sources – fichiers stockés sur OneDrive, SharePoint, bases de données en ligne Salesforce, base de données Azure, l'actualisation est le plus souvent automatique (toutes les heures par exemple pour des fichiers stockés sur OneDrive).

Il est également possible d'aller chercher uniquement les données ajoutées ou modifiées depuis la précédente actualisation : l'actualisation incrémentielle repose sur l'existence d'une date témoin, et accélère considérablement le temps d'actualisation d'un rapport. Pour plus d'information sur ce point, je vous invite à lire l'article que je lui ai consacré : https://daxone.fr/un-choix-parmi-les-nouveautes-de-fevrier-2020/.

B. Se connecter

Power BI propose une vaste bibliothèque de connecteurs – et encore celle-ci évolue avec chaque nouvelle livraison du logiciel : les décrire tous serait fastidieux, mais il est possible de dégager les grandes lignes et le fonctionnement des principaux connecteurs.

Vous trouverez sur le site de l'éditeur une description très complète de la plupart des connecteurs : https://docs.microsoft.com/fr-fr/power-bi/desktop-data-sources

1. Les trois types de connexion

Un premier choix essentiel consiste à définir le type de connexion : import ou direct.

- L'option la plus fréquente consiste à **importer** les données : une fois la requête lancée, Power BI récupère et stocke en local l'ensemble des données.
  - L'intérêt majeur de ce type de connexion repose sur les performances de Power BI lors de la création de rapports, les possibilités de modélisation et la réactivité aux actions de filtrage de l'utilisateur.
  - L'inconvénient tient à ce que le poste sur lequel est conçu ou utilisé le rapport doit avoir une puissance suffisante, en termes de RAM (pour le stockage des données) et de CPU (pour les calculs).
  - Un autre inconvénient possible est la limite de taille du fichier : 1 Go pour la version standard, 10 Go pour les versions Pro et Premium.
  - Enfin, ce type de connexion implique une actualisation manuelle ou programmée des données – par opposition à la visualisation de données en temps réel.

Malgré ces quelques inconvénients, l'import des données est le mode à privilégier.

- Il est également possible de se connecter en **direct** à certaines sources (bases de données).
  - L'intérêt ici est d'avoir en permanence des rapports à jour, sans avoir besoin d'actualiser les données. Une simple action (filtre ou sélection dans le visuel) suffit à mettre à jour les données. Par ailleurs, la puissance du poste n'est plus un critère limitant.
  - En revanche, un inconvénient tient à la capacité de la source : il faut qu'elle soit suffisamment puissante pour accepter une connexion directe. Les performances de l'application peuvent s'en ressentir.
  - Par ailleurs, toutes les sources ne sont pas disponibles.
  - Les possibilités de modélisation des données sont limitées (puisqu'elles ont déjà été définies dans la source).
  - Enfin d'autres limitations incluent la restriction de certaines fonctionnalités de transformation et de création de formule (notamment les fonctions Time Intelligence).

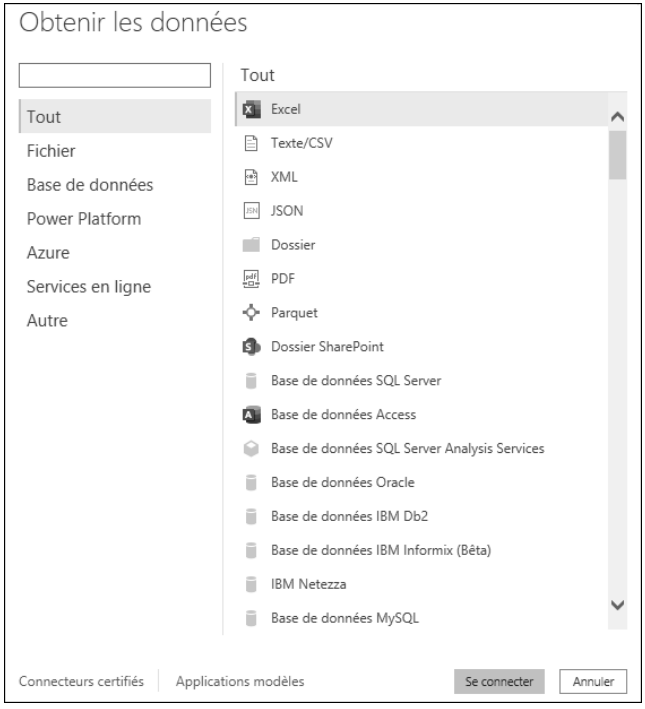
Il existe deux types de connexion en direct : la connexion DirectQuery et la connexion directe. La première est plus générale, mais aussi moins performante, la seconde concerne exclusivement SSAS, Azure ou Power BI Service – elle est plus performante dans la mesure où les données sont déjà stockées sur le modèle qu'utilise Power BI.

Découvrir les types de connexion

- Lancez Power BI Desktop et cliquez sur **Obtenir les données**.

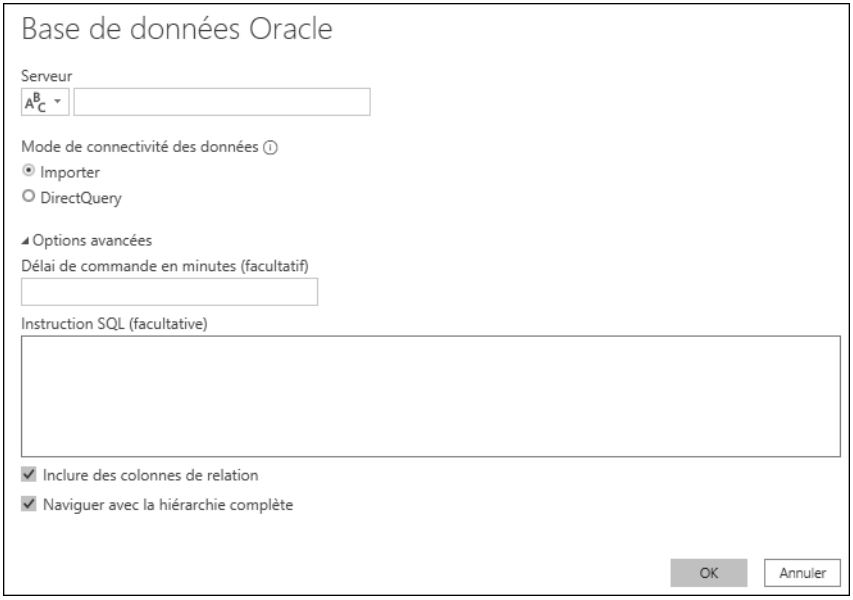
*Vous trouvez aussi la fonctionnalité **Obtenir les données** sur l'onglet **Accueil**.*

- Pour accéder à une source, sélectionnez le connecteur dans la liste, puis cliquez sur **Se connecter**.

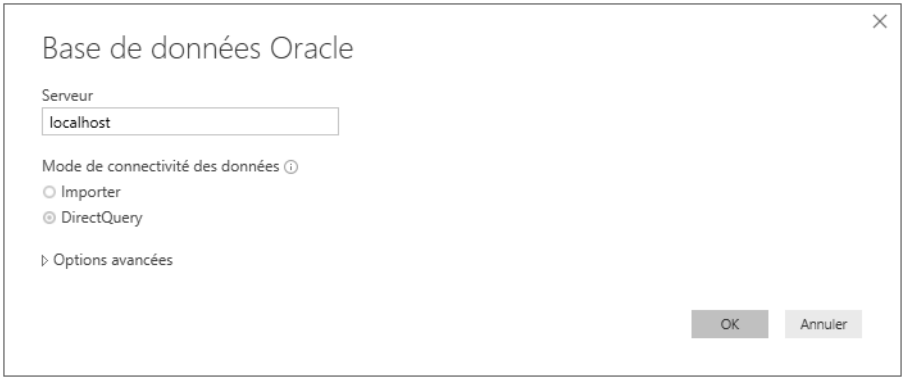


En fonction du connecteur choisi, des options apparaissent, le nom de la base, le mode de connectivité.

Voici quelques exemples de connecteurs à des bases de données Oracle et SQL Server :



Connexion avec import sur une base Oracle : dans les options avancées, remarquez la possibilité de saisir directement du code SQL



Connexion DirectQuery sur une base Oracle

A. Introduction

Le rôle central du langage DAX dans l'analyse des données n'est plus à démontrer : aussi bien pensée que soit votre source de données, aussi bien construit que soit votre modèle, il ne s'agit que des pierres sur lesquelles vous allez pouvoir fonder vos analyses, et celles-ci seront toujours basées sur des formules DAX.

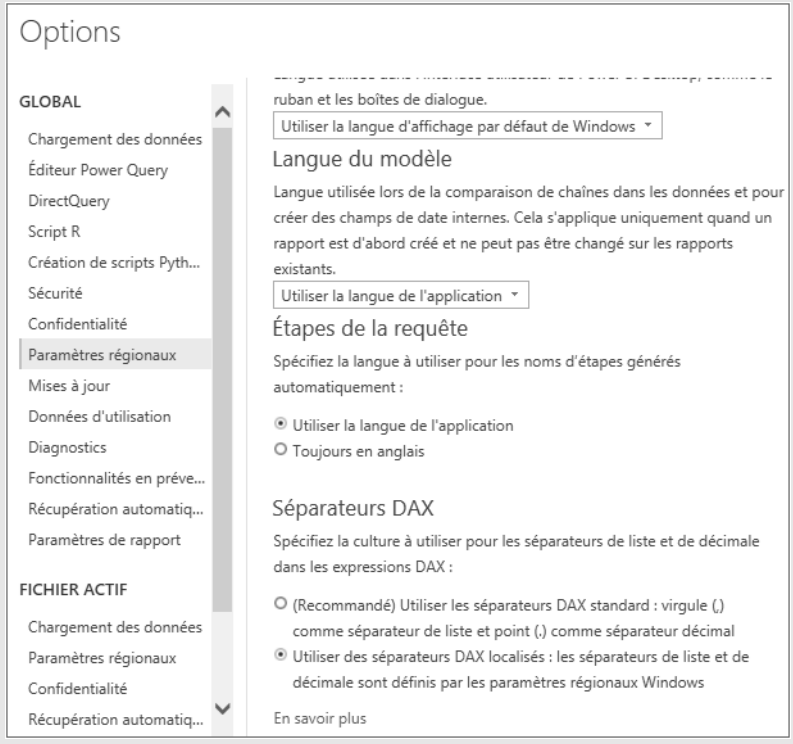
Ce chapitre est centré sur LA fonction DAX par excellence : CALCULATE.

Parce que c'est la fonction pivot, celle qui demande de maîtriser les deux notions clés du DAX, à savoir les fonctions de table et le contexte de filtre et, par conséquent de maîtriser toute une batterie d'autres fonctions considérées à juste titre comme centrales.

Et parce que derrière une apparente simplicité – après tout, CALCULATE se contente de modifier le contexte de filtre avant de calculer une expression -, se cache un mécanisme d'une richesse et parfois d'une complexité remarquables.

Après un préambule destiné à poser les bases du travail (rappel des notions clés, debugging, validation d'un calcul), nous aborderons CALCULATE, nous évoquerons des modèles de formule, nous entrerons dans les arcanes de la fonction, et nous verrons comment CALCULATE permet de réaliser des analyses courantes.

Depuis la mise à jour de mai 2020, les séparateurs DAX sont par défaut la virgule pour séparer les listes et les arguments au sein d'une fonction, et le point comme séparateur décimal. Un système « à l'américaine », donc, par opposition aux paramètres localisés, le point-virgule pour séparer les arguments et virgule comme séparateur décimal, pour ce qui concerne la France. Nous restons dans ce livre fidèle à ces derniers. Vous pouvez passer de l'un à l'autre à l'aide du menu ***Fichier - Options et paramètres - Options****, puis dans la rubrique* ***Global - Paramètres régionaux****, faites votre choix des* ***séparateurs DAX*** *:*



Si vous préférez utiliser les (nouveaux) paramètres recommandés par Power BI, il vous suffira, dans tous les exemples de formules qui suivent, de remplacer le point-virgule par la virgule.

B. Préambule

Cette section traite d'un ensemble de points fondamentaux et généraux de DAX. Vous n'y trouverez donc pas d'exercices d'application, mais je vous encourage à appliquer ces concepts et règles dans toutes vos formules DAX. Il est essentiel que vous lisiez très attentivement les pages qui suivent. Et peut-être, que vous les relisiez par la suite !

1. Les mesures, leur format, leur nom

Pour l'essentiel, les formules doivent être utilisées dans le cadre de la création de mesures, et plus rarement pour la création de colonne.

Rappelons en effet que les mesures ne prennent pas de place dans votre modèle (n'utilise pas d'espace de stockage), et garantissent donc un document plus léger et plus performant. Rappelons que les mesures ne sont calculées qu'au moment de leur utilisation, selon le contexte dans lequel elles apparaissent, et pas pour chaque ligne de la table dans laquelle, à l'inverse, une colonne est « physiquement » créée.

Rappelons également que toute donnée numérique issue de la source et destinée à être analysée (montant des ventes, quantité, résultat de test, etc.), doit être masquée, et remplacée par une mesure équivalente (`[Montant] = SUM(Ventes[Montant des ventes]`). Nous verrons plus loin le lien avec CALCULATE.

Il est par ailleurs utile, lors de la création de formule, de prendre le réflexe de formater tout de suite la mesure nouvellement créée. Enfin pour ce qui concerne le nom de la mesure, évitez les lettres accentuées dans les quelques premiers caractères (afin qu'Intellisense retrouve rapidement votre mesure).

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 généré automatiquement lors de la création d'une colonne par le biais d'une formule, ainsi que lors de l'utilisation d'une fonction de type itérateur (en particulier les fonctions X – SUMX, etc. – et la fonction FILTER).

Le contexte de filtre, lui, 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.

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 bi-directionnelle, permettent 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. Nous reviendrons sur cette notion capitale.

3. Mise en forme des formules

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.
- Ajouter autant de commentaires que nécessaire, précédés d'un //.
- Faciliter la vue des parenthèses ouvrantes et fermantes.
- Aller à la ligne autant de fois que vous le voulez.

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

Notez aussi qu'une variante, notamment dans le cadre d'un CALCULATE, peut consister à décaler le point-virgule, pour mieux être en mesure de mettre en commentaire une des lignes du code :

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

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

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

4. Raccourcis de l'éditeur de formule DAX

L'éditeur de formule fourmille de raccourcis clavier permettant d'accélérer la saisie ou de faciliter la correction des formules DAX.

Avec un minimum de pratique, vous gagnerez un temps précieux pour remettre en forme, corriger à plusieurs endroits en même temps, intervertir les lignes, etc.

Voici quelques-uns de ces raccourcis.

Les cinq raccourcis-clavier les plus utiles

- Sélectionner toutes les occurrences du terme (ou des caractères) actuellement sélectionné (exemple : une colonne, une table ou une fonction qu'il faut remplacer partout dans la formule).

  Ctrl F2 (ou Ctrl ⇧ L)
- Mettre en commentaire (ou à l'inverse, enlever la mise en commentaire)

  Sélectionner la ou les lignes et Ctrl /
- Aller à la ligne avec indentation (retrait)

  ⇧ ↵

- Copier une ligne au-dessus/en-dessous
  ⇧ Alt ↑ ou ↓
- Valider une proposition de fonction d'Intellisense
  ⇥

Manipuler les lignes

- Déplacer une ligne vers le haut/le bas
  Alt ↑ ou ↓
- Insérer la ligne ci-dessous
  Ctrl ↵
- Insérer la ligne ci-dessus
  Ctrl ⇧ ↵
- Sélectionner la ligne actuelle
  Ctrl I
- Naviguer jusqu'à une ligne en indiquant son numéro
  Ctrl G (taper ensuite ↵ pour masquer la zone de saisie du numéro de ligne)

Saisir du texte

- Insérer plusieurs curseurs aux endroits choisis en cliquant (utile pour saisir un même texte à plusieurs endroits)
  Alt Clic
- Activer plusieurs curseurs au même endroit sur plusieurs lignes
  Ctrl Alt ↓ ou Ctrl Alt ↑
- Sélectionner toutes les occurrences de la sélection actuelle (exemple : une colonne, une fonction qu'il faut remplacer partout).
  Ctrl ⇧ L ou Ctrl F2
- Sélectionner le mot entier (là où se trouve le curseur) (Si une portion de texte a déjà été sélectionnée, Ctrl D sélectionne la prochaine occurrence de la même sélection, et ainsi de suite : cela permet de modifier l'ensemble des portions sélectionnées d'un coup)
  Ctrl D

Commentaire et indentation

- Mettre en commentaire (ou l'inverse)
  Sélectionner la ou les lignes et Ctrl /
- Augmenter l'indentation / réduire l'indentation
  ⇥ / ⇧ ⇥
- Aller à la ligne sans indenter
  Alt ↵
- Aller à la ligne avec indentation
  ⇧ ↵

Utiliser Intellisense

- Valider une proposition
  ⇥
- Pour rappeler la fenêtre Intellisense
  Alt I

5. Correction de formule avec les variables

Il n'existe pas, dans Power BI, de moteur d'analyse des formules permettant d'en arrêter l'exécution au moment où une erreur est détectée (comme c'est le cas dans Excel notamment). Pour des formules complexes, longues, ceci peut être un problème.

C'est là qu'une utilisation astucieuse des variables peut vous aider.

Les variables sont déclarées, et exécutées, au début du script. Elles sont couramment utilisées pour n'exécuter un calcul qu'une fois, même si celui-ci est appelé plusieurs fois dans la formule. Leur première utilité est donc d'améliorer la performance de la formule.

Leur deuxième raison d'être est d'améliorer la lisibilité de la formule.
Leur troisième intérêt est d'effectuer un calcul sur l'état du contexte de filtre au début de l'exécution de la formule (en effet, le calcul n'étant effectué qu'une fois, la valeur de la variable est figée pendant toute la durée de la formule). Ce point, important, est développé à la fin de cette section.

Pour illustrer ce point, je me sers d'un exemple proposé par Marco Russo dans un tutoriel de Guy In A Cube : https://www.youtube.com/watch?v=9SV2VnYbgg4

Nous allons commencer par voir les trois états de mise en forme de la formule : d'abord sans mise en forme, puis avec, et enfin avec l'introduction des variables.

Cette formule retourne une liste de couleurs dont le nombre varie en fonction du choix fait par l'utilisateur à l'aide d'un segment *Couleurs_seg*.

Premier état :

```
Couleurs =
IF (
   COUNTROWS ( DISTINCT ( 'Produit'[Couleur] ) ) > SELECTEDVALUE
( 'Couleurs_seg'[Nombre de Couleurs] );
   CONCATENATEX ( TOPN ( SELECTEDVALUE ( 'Couleurs_seg'[Nombre de
Couleurs] ); VALUES ( 'Produit'[Couleur] ));'Produit'[Couleur]; " , ")
& " et plus...";
   CONCATENATEX ( DISTINCT ( 'Produit'[Couleur] ); 'Produit'[Couleur];
" , " )
)
```

Vous remarquerez tout de suite que le code est très confus, donc difficile à lire, et par conséquent difficile à corriger.

Et en effet, ce code renvoie une erreur. Mais où se situe-t-elle ?

Le premier nettoyage consiste à formater le code selon les règles courantes, par exemple en le passant à la moulinette de www.daxformatter.com.

A. Introduction

Avec ce nouveau chapitre, nous entrons de plain-pied dans Power Query et ses capacités à nettoyer et à transformer les données issues de la source.

Nous allons voir le fonctionnement général de Power Query, et l'articulation avec les lignes de code M.

Nous aborderons le travail au niveau des requêtes (ou tables, les termes sont synonymes), au niveau des colonnes (modifier, supprimer, ou ajouter des colonnes), la transformation des champs de type texte, numérique ou date et le traitement des tableaux croisés.

La plupart du temps, nous verrons que les opérations peuvent être réalisées à l'aide de l'interface graphique, mais aussi à l'aide du code M, et pourquoi comprendre ce code et en connaître certaines syntaxes simples permet d'accélérer le travail, et souvent de dépasser les limites de l'interface.

Les opérations que nous illustrons dans ce chapitre sont courantes, des opérations que tout utilisateur de Power Query doit connaître.

Le chapitre Exploiter la puissance de Power Query vous permettra de pousser plus loin le travail à l'aide de l'interface, avec entre autres la mise en place de tables d'agrégation, ou encore de paramètres.

Le chapitre Guide pratique du code M propose, lui, une approche plus globale de ce code, avec des éclairages sur sa structure, ses entités et ses concepts, et illustrera ces différents points à l'aide d'un ensemble de cas de mises en œuvre fréquents.

B. Fonctionnement général

L'accès à Power Query est possible à tout moment du travail dans Power BI : soit à partir du navigateur que nous avons vu au chapitre précédent (options **Transformer les données**), en d'autres termes immédiatement après la phase de connexion aux données, soit à partir de Power BI (bouton **Transformer les données** dans le ruban **Accueil**), en cours d'élaboration du rapport.

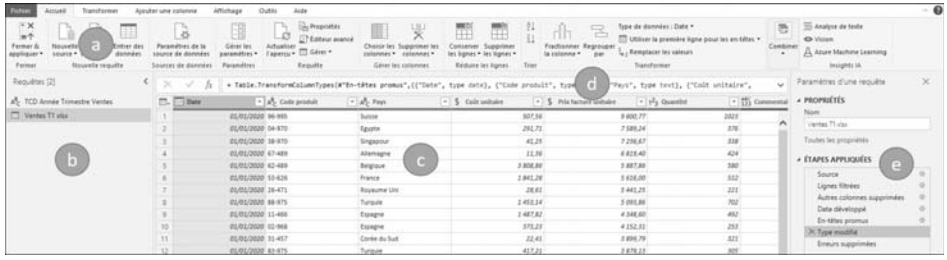
Il faut ici distinguer les sources locales (ou personnelles) et les sources d'entreprise (base de données notamment, mais aussi toute source partagée) : les premières nécessitent le plus souvent un travail de nettoyage et de préparation, ce qui est moins vrai des secondes. Mais il n'existe aucune règle définitive en la matière.

Il est utile de rappeler que le travail dans Power Query est effectué sur un échantillon de données (1000 lignes) : sur la base de cet échantillon, vous allez pouvoir définir un ensemble d'opérations (dont le nom précis est « étapes »). C'est seulement lorsque vous serez satisfait par cet ensemble d'étapes que vous allez envoyer la requête à la source, qui ramènera alors l'intégralité des lignes de la source.

Autrement dit, tout travail dans Power Query se conclut par l'activation de la fonctionnalité **Fermer & appliquer** : c'est à ce moment que vous envoyez la requête.

Il faut souligner un autre aspect de Power Query : toute étape de transformation correspond à une ligne de code M, visible dans la barre de formule. Nous allons y revenir dans la section suivante.

Pour rappel, un aperçu de l'interface graphique de Power Query :



a : l'ensemble de rubans, et dans le ruban **Accueil** la fonctionnalité **Fermer & appliquer** ;

b : les requêtes, ou tables (mais aussi, nous le verrons plus loin, les fonctions personnalisées, les paramètres, etc.).

c : la table active.

d : la barre de formule ; si vous ne voyez pas la barre de formule, rendez-vous dans le ruban **Affichage** et cochez la case **Barre de formule**.

e : les étapes.

C. Lire et comprendre le code M

1. Aperçu de la structure d'un code M

Dans ce chapitre, nous allons souvent alterner entre l'interface graphique et le code M : dans la présente section, je vous propose une petite introduction à ce code, sa structure, et comment l'afficher, mais uniquement dans le but de vous proposer par la suite des petites modifications souvent très simples, en général à partir de la barre de formule – comme vous pourriez le faire dans Excel.

Voici à quoi ressemble un code M résultant de transformations courantes. Les étapes apparaissent en gras dans le code ci-dessous (chacune des étapes correspond à une transformation). Les retours à la ligne ont été ajoutés dans un souci de clarté. Ce code a été généré automatiquement par l'interface graphique.

En deux mots : il s'agit d'abord d'aller chercher les données (étapes **Source** et **Feuil1 Sheet**), puis d'opérer une série de transformations (de l'étape **En-têtes promus** à **Lignes filtrées**), et enfin de retourner le résultat (après le **IN**) :

```
let
   Source =
Excel.Workbook(File.Contents(ChoixSource), null, true),
   Feuil1_Sheet =
Source{[Item="chine",Kind="Sheet"]}[Data],
   #"En-têtes promus" =
Table.PromoteHeaders(
      Feuil1_Sheet,
      [PromoteAllScalars=true]),
   #"Type modifié" =
Table.TransformColumnTypes(
      #"En-têtes promus",
      {{"date", type date},
      {"heure", type time},
      {"confirmés", Int64.Type}}),
      #"Index ajouté" =
Table.AddIndexColumn(#"Type modifié", "Index", 1, 1),
   #"Personnalisée ajoutée" =
Table.AddColumn(
      #"Index ajouté",
      "Date heure",
      each [date] & [heure]),
   #"Lignes filtrées" =
Table.SelectRows(
      #"Personnalisée ajoutée",
      each ([date] <> null))
in
   #"Lignes filtrées"
```

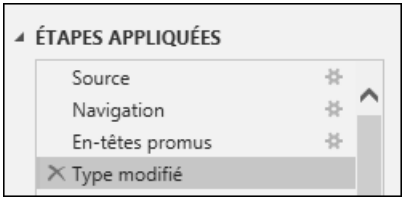
De manière plus générale (et peut-être plus lisible), la structure du code est le plus souvent la suivante :

```
let
   Source = <nom de la source> ,
   // cette ligne est un commentaire
   #"Etape 1" = FonctionM (Source, paramètres) ,
   #"Etape 2" = FonctionM (#"Etape 1", paramètres) ,
   #"Etape finale" = FonctionM (#"Etape 2", paramètres)
in
   #"Etape finale"
```

Le mot-clé LET (en minuscules) permet d'énoncer une série d'opérations tandis que le mot-clé IN (en minuscules) indique ce qui ressortira. C'est le plus souvent la dernière étape, mais pas nécessairement : il est possible d'invoquer une étape antérieure, par exemple pour corriger le code.

Comme vous pouvez le constater, l'élément clé est **l'étape**.

Elle porte un nom et appelle une fonction à laquelle elle passe des paramètres. Dans l'interface de Power Query, vous retrouvez le nom des étapes dans la rubrique **ÉTAPES APPLIQUÉES** du volet **Paramètres d'une requête** :



Dans cet exemple, quatre étapes ont été appliquées

Vous retrouvez le code dans la barre de formule :

Si le nom de l'étape est un seul mot, il apparaît tel quel : *Source*, *Feuil1_Sheet*. S'il est composé de plusieurs termes, il apparaît encadré de guillemets et précédé d'un dièse : *#"Etape 2"*.

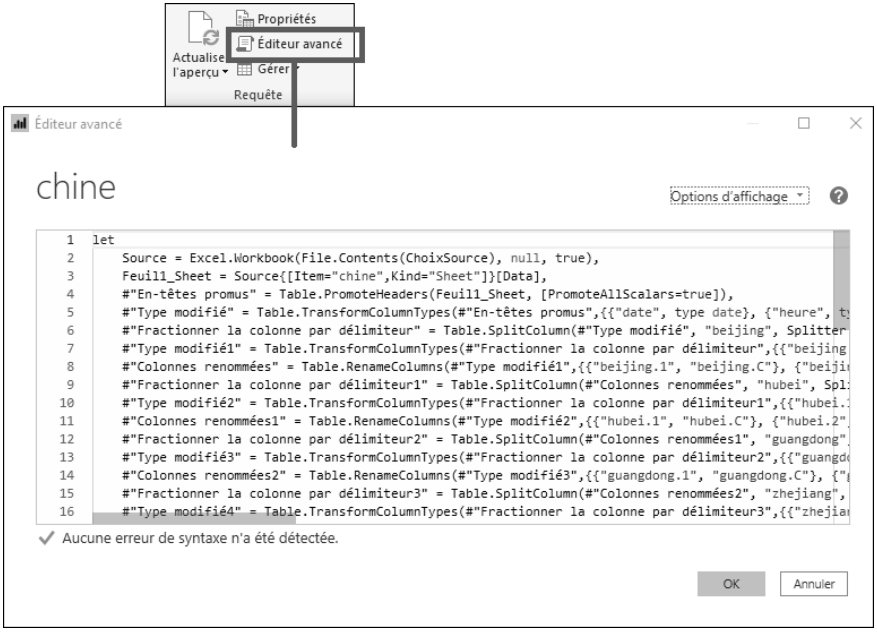
Attention, M est sensible à la casse : il faut impérativement respecter les minuscules et les majuscules (le plus souvent au début du nom de la fonction).

Autres remarques sur le code visible ci-dessus :

- Parmi les paramètres d'une fonction, vous retrouvez quasi systématiquement une référence à une autre étape, le plus souvent celle qui précède.
- Pour insérer un commentaire, faite précéder la ligne d'un double slash //.
- Pour insérer un commentaire sur plusieurs lignes, vous pouvez aussi utiliser la syntaxe /* ... */.

2. Afficher et éditer le code M

Bien que vous puissiez éditer le code directement dans la barre de formule, la façon la plus efficace d'afficher et de modifier le code M est d'utiliser l'éditeur avancé (accessible depuis l'onglet **Accueil**), fenêtre dans laquelle l'intégralité du code (c'est-à-dire toutes les étapes) est visible :



Plutôt que d'un éditeur avancé, il faudrait plutôt parler d'un éditeur rudimentaire pour utilisateur avancé ! En effet, vous n'y trouverez que peu de coloration syntaxique (mais tout de même bien utile), et une aide à l'écriture (M IntelliSense).

Moyennant quoi les codeurs **experts** utilisent souvent Notepad++ avec l'extension M de Lars Schreiber, ou encore Visual Studio (version 2017 et suivantes) avec le SDK Power Query.

Ce serait trop pour notre approche du M, aussi je me contenterai dans la suite de cet ouvrage de la barre de formule ou de l'éditeur avancé.

En termes de coloration syntaxique, le rouge indique une zone de texte (avec ses guillemets), le bleu turquoise un type de donnée, le bleu foncé un mot-clé. Tout le reste est noir.

D. Transformer les colonnes

Dans chacune des sections de ce chapitre, nous allons reprendre les fichiers que nous avons commencé à créer au chapitre précédent. Vous pouvez repartir de vos fichiers ou les retrouver dans le répertoire ***exemples\chapitre3*** *disponible en téléchargement.*

Transformer et nettoyer les colonnes sont les premières actions que vous réaliserez dans Power Query : cela inclut notamment supprimer les colonnes ou les lignes inutiles et vérifier ou changer le type de la colonne.

1. Conserver ou supprimer les colonnes

Dans cette section, nous utilisons le fichier ***source02.pbix****.*

Il existe une règle générale dans Power BI : ne ramenez que les données dont vous avez besoin. La quantité de données a un impact sur la performance de l'application, en lien avec les capacités de votre PC. Plus le fichier est léger, mieux Power BI réagira.

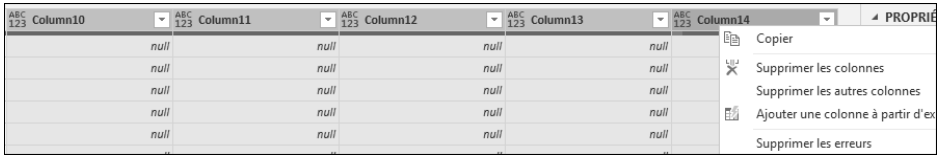
Conserver uniquement les colonnes nécessaires est donc la première action que vous devez faire. Cela suppose bien sûr que vous ayez une bonne idée de l'objectif que vous fixez à votre rapport. Mais rien n'est non plus définitif : vous pouvez à tout moment rajouter des colonnes ou annuler la suppression d'une colonne.

a. Conserver ou supprimer les colonnes à l'aide de l'interface

- Ouvrez le fichier **source02.pbix**.
- Dans le ruban **Accueil** - groupe **Requêtes** cliquez sur **Transformer les données** pour lancer Power Query.
- Dans le volet **Requêtes**, cliquez sur **Ventes** pour faire apparaître la table.

Sur la droite de la table, des colonnes superflues (**Column10** à **Column14**) ont été extraites : nous allons les supprimer.

- En maintenant la touche Ctrl enfoncée, cliquez un à un sur les en-têtes des cinq colonnes concernées.
- Puis effectuez un clic droit pour ouvrir le menu contextuel et cliquez sur **Supprimer les colonnes**.



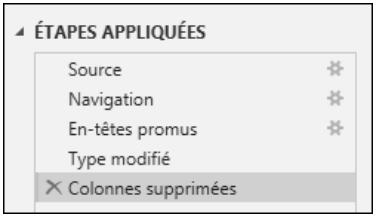
Les colonnes disparaissent, et une nouvelle étape (**Colonnes supprimées**) a été créée dans le volet **Paramètres d'une requête**. Le nom de l'étape est précédé d'une croix, qui permet la suppression et le retour à l'état avant transformation.

Il existe différentes manières de supprimer une ou plusieurs colonnes :

- par clic droit, ainsi que nous venons de le voir,
- à l'aide de la fonctionnalité **Choisir les colonnes**,
- en sélectionnant les colonnes à conserver, plutôt que celles à supprimer : lorsque vous supprimez plus de colonnes que vous n'en conservez, cette option peut être plus rapide. Cette opération peut être effectuée par clic-droit (option **Supprimer les autres colonnes** du menu contextuel), ou à l'aide de la fonctionnalité **Supprimer les autres colonnes** du menu déroulant **Supprimer les colonnes**).

Voyons par exemple la deuxième méthode :

- Dans le volet **Paramètres d'une requête**, cliquez sur la croix à gauche de l'étape **Colonnes supprimées**. Ceci annule l'étape, et les colonnes réapparaissent donc.



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"
)
```

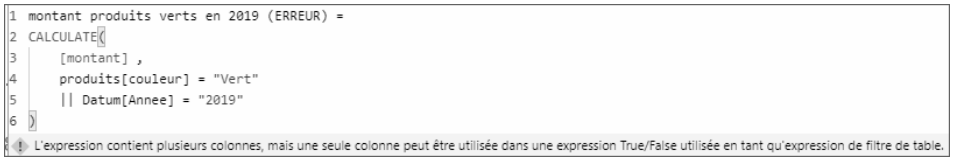
201810	€ 5 928,00	
201811	€ 6 338,00	
201812	€ 4 270,00	
201901	€ 8 621,00	924,00 €
201902	€ 8 278,00	386,00 €
201903	€ 7 344,00	330,00 €
201904	€ 5 420,00	258,00 €

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.