Chapitre 3
Power Query Online

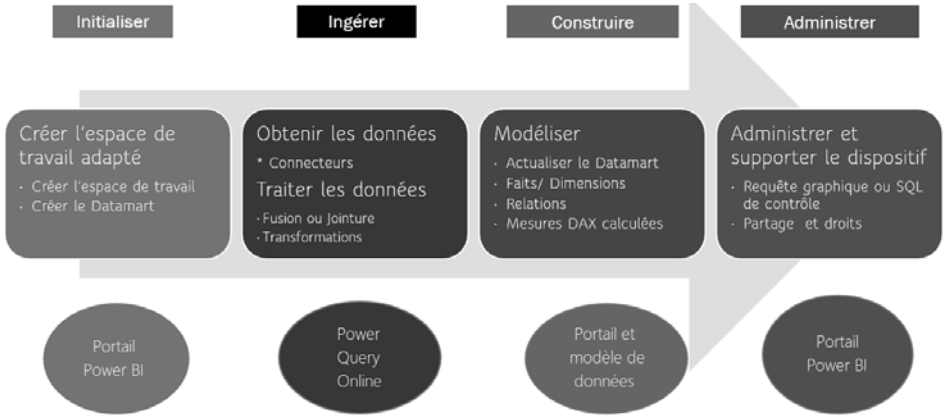
1. Principes de base

1.1 Utilité et plus-value de Power Query Online

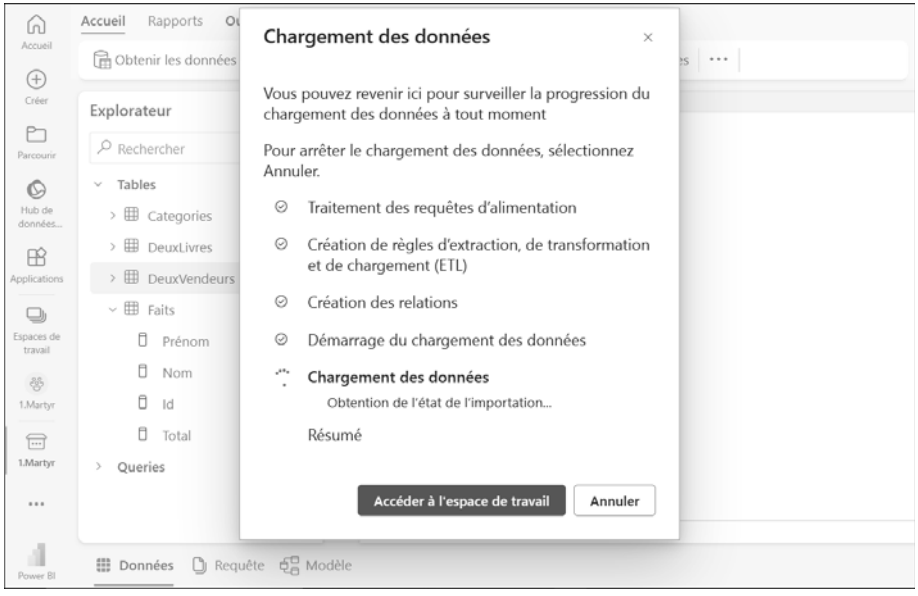
Power Query Online (PQO) est le module d'extraction de transformation et de chargement (ETL - *Extract Transform and Load*) des données du datamart, mais aussi de nombreuses applications Microsoft : Microsoft Fabric, Power Apps, Power Automate, Azure Data Factory, Dynamics 365 Customer Insights, les Dataflows et Excel pour Mac. Le fait de mutualiser un même outil dans plusieurs solutions permet à Microsoft de proposer un outil mieux développé et mieux testé.

Chaque version de PQO a une technologie de stockage différente. Ainsi, Power Query Online de datamart Power BI permet des sources variées, mais un stockage de données final disponible uniquement dans une base SQL Server alors que d'autres vont stocker en dataverse ou en data lake.

Concrètement, Power Query Online est un outil d'ingestion qui permet de se connecter aux sources de données, d'incorporer des données et de les préparer, c'est-à-dire de nettoyer les données et d'ajouter des colonnes supplémentaires si nécessaire pour constituer un magasin de données propres en libre-service avec des droits d'accès.



Comme en atteste le tout dernier écran à l'enregistrement de Power Query Online figurant ci-dessous, PQO alimente les données sources d'abord, crée et vérifie des règles de transformation avant de transformer ou de faire transformer les données puis termine enfin par le chargement sous forme d'un stockage en base puis d'une importation dans le datamart. Cet import permet une grande vitesse d'exécution, puisqu'il est en mémoire sur le serveur du datamart.



© Editions ENI - All rights reserved

Le cas du Query Folding que nous verrons dans ce chapitre, section Le Query Folding et ses indicateurs, opère quelques transformations simples en amont à la source mais, in fine, la transformation finale se fait bien dans l'interface PQO, avant le chargement complet dans le datamart et son actualisation.

La force de PQO et de son langage M est de savoir :

- ajouter des sources de données avec des connecteurs déjà testés et éprouvés sur de très nombreuses sources (dans les Dataflows, soit bien avant l'existence des datamarts),
- ajouter des tables et des colonnes (à noter : DAX ne peut pas réaliser ces actions dans les datamarts),
- filtrer ou supprimer des lignes et des colonnes, supprimer ou remplacer les erreurs,
- inclure des requêtes SQL dites « natives » directement à l'intérieur de Power Query ou du code M,
- recopier vers le bas les blancs dans les sources mal structurées (tableau croisé Microsoft Excel),
- changer les types des colonnes et ainsi impacter la compression de taille de colonne et le datamart,
- transformer les colonnes de texte (par exemple découper les chaînes de caractères, remplacer des caractères ligne à ligne, aider au typage logique et changer les types),
- nettoyer les données (réaligner les majuscules et minuscules de façon homogène, uniformiser du texte en majuscules, remplacer les caractères non imprimables),
- simplifier le modèle de données par la combinaison de requêtes pour ne créer qu'une seule table de fait,
- supprimer les doublons de table ou de colonne avec les fonctions `Table.Distinct` ou `Column.Distinct` (fonctions natives du ruban),
- remplacer les erreurs de table ou de colonne (`Table.ReplaceErrorValues` et `Column.ReplaceErrorValues`),

– saisir des paramètres Power Query pour gérer des scénarii de calcul, réutiliser les requêtes ou des formules dans plusieurs contextes différents avec le même code,
– créer des fonctions ad-hoc pour automatiser les traitements personnalisés.

Certaines fonctionnalités sont nettement supérieures à celles de la version Power Query de Power BI Desktop :

– Les indicateurs de Query Folding montrent clairement ce qui est optimisé ou pas dans les étapes.
– Le Plan de Requête est un plus qui détaille au serveur source les requêtes passées ou non ainsi que leur ordonnancement et leur blocage.
– Une version française souvent mieux traduite que Power Query de Power BI Desktop : par exemple le type est « Monnaie » pour les devises (*currency*) en PQO au lieu de « Nombre décimal fixe » en Power Query Desktop.

1.2 Faiblesses de Power Query Online

Voici les faiblesses de Power Query (versus DAX et les modèles) :

– PQO fait ses transformations au chargement et non au fil de l'eau ni dans un train de traitement conditionnel. Il va calculer toutes les colonnes au chargement ligne par ligne, ce qui prend du temps au premier chargement (dans le moteur d'analyse en ligne).
– PQO n'est pas adapté pour préparer des mesures et des indicateurs tout faits pour que les utilisateurs puissent les consommer par-dessus le modèle sémantique, c'est le rôle de DAX.
– PQO s'avère lent sur les fonctions de recherche de valeur parmi une liste, ce qui peut évoluer à terme.
– PQO s'avère également lent au premier chargement sur les fusions ou jointures entre très grandes tables avec un grand nombre de lignes (supérieur à 500 000).

© Editions ENI - All rights reserved

1.3 Utiliser Power Query Online ou DAX ?

Contrairement à Power BI Desktop, certaines opérations sont impossibles en DAX, comme les fonctions DAX de création de table ou d'ajout de colonne : le menu **Créer une Table DAX** n'existe pas dans un datamart, ce qui simplifie la question DAX ou PQO. Cependant, tout comme dans Power BI Desktop, bon nombre d'opérations sont communes entre DAX et PQO dans Power BI datamart, et il est important de comprendre dans quel cas utiliser l'un ou l'autre.

Préférez Power Query Online	**Préférez DAX**
Renommage de champ	Création de mesures, pour opérer des calculs ou des statistiques
Transformation de texte	Pour récupérer la sélection de l'utilisateur
Envoi de requête au serveur source	Pour calculer des agrégations ou des statistiques en fonction de l'usage et du contexte de filtrage
Pour reproduire ou se référer à la source de données	Pour éviter de faire des jointures ou des fusions en table, ce qui peut s'avérer très lent
Pour le typage de donnée	Pour des formules complexes ou lourdes, comme les formules financières
Pour créer une table de mesure et les regrouper ainsi au même endroit permettant ainsi de les retrouver et documenter facilement	Pour écrire chaque mesure plutôt qued'ajouter des colonnes calculées qui vont ralentir le rafraîchissement. Vous pourrez regrouper ces mesures dans des « dossiers » par thème dans le modèle de données

Préférez Power Query Online	Préférez DAX
Pour tout ce qui doit se faire ligne à ligne par exemple pour du nettoyage une suppression de doublons ou d'erreur ou un tri des données	Pour tout ce qui doit se synthétiser en multiligne
Pour ajouter une colonne de rang dans une table de synthèse à gros grain (pour classer l'intégralité des vendeurs ou des produits un par un directement dans PQO)	Pour faire un top 10 des ventes par produit ou un top 5 des vendeurs, DAX sera plus performant et plus pratique. L'outil n'aura pas à classer tous les produits ou tous les commerciaux au-delà du seuil de 10 ou 5, ni à exiger une table PWO dans un petit tableau de synthèse à regroupement
Pour le tri simple ou complexe	Pour la fonction conditionnelle `SWITCH`, qui n'existe pas en PQO

Tout est une affaire de cas pour les fonctions présentes des deux côtés. Ainsi, pour calculer les rangs, on s'aperçoit que le *ranking* Power Query fonctionne bien sur des tables groupées, car il y a une synthèse avec unicité sur le groupe. En revanche, si vous avez des tables de fait et des tables de dimension bien établies, le *ranking* sera impossible dans Power Query Online, même si le menu existe : la fonction PQO **Colonne de rang** que vous verrez dans la section Établir un rang sur une table de ce chapitre demande de classer les éléments de la dimension avec une valeur numérique qui n'existe pas en dimension, c'est donc impossible. Quant à la table de fait, le rang se ferait commande par commande et non commercial par commercial ou produit par produit, ce qui offre peu d'intérêt.

© Editions ENI - All rights reserved

1.4 Outil externe

Il est pratique d'avoir un outil pour stocker tous les scripts PQO à un seul endroit, et de disposer d'un éditeur pour les requêtes qui sache reconnaître le format et identifier les anomalies. L'outil qui est le plus approprié est **Visual Studio Code**. Les instructions de mise en place et le lien sont indiqués au chapitre Initialiser votre datamart, section Conseils de mise en place et de conception - Outils complémentaires utiles.

Remarque

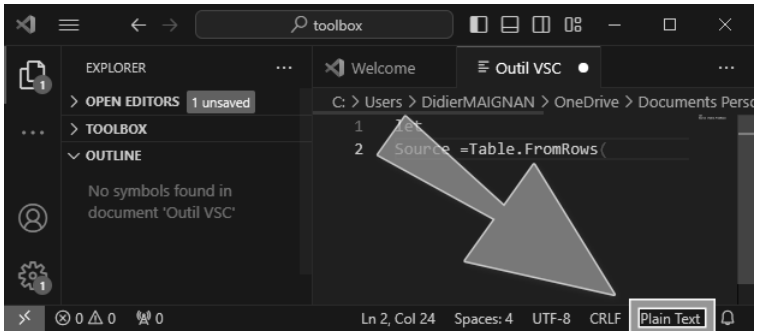
Pour que Visual Studio Code soit efficace, il faut copier la totalité du code M depuis l'éditeur avancé PQO, sinon il ne pourra pas vous corriger.

Pour choisir le format Power Query / M dans **Visual Studio Code (VSC)**, il faut remplacer **Plain Text** en bas à droite et taper Power ou Power Query, ainsi vous pourrez sélectionner **Power Query Formula Language**.

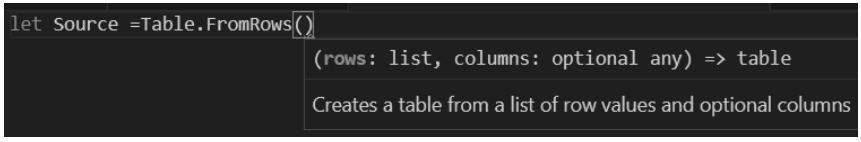
Voici un exemple d'aide de VSC sur Power Query Online en entrant le début du code (cf. fichiers en téléchargement de cet ouvrage, référence code M 3.1) :

```
let
Source =Table.FromRows(
```

Après avoir installé l'extension Power Query Formula Language, comme indiqué dans le chapitre Initialiser votre datamart - section Conseils de mise en place et de conception - Outils complémentaires utiles, remplacez **Plain Text** en bas à droite par **Power Query Formula Language**, comme indiqué dans l'image ci-dessous.



Dès la première parenthèse gauche, derrière le code `FromRows`, l'éditeur VSC propose les arguments, alors que l'éditeur natif n'affiche rien.



Vous pouvez découvrir le formatage couleur automatique de VSC par vous-même avec le petit bout de code M suivant (cf. référence code M 3.2) présent dans les fichiers M de la toolbox du livre, au nom de 3.2 InfoQuatreVendeurs.pq :

```
// InfoQuatreVendeurs du 21/07/2023
let Source =Table.FromRows(
    {
        {1, "Olivier", "06 84 45 41"},
        {2, "Jean", "09 87 65 42"},
        {3, "Eulalie", "09 87 65 43"},
        {4, "André", "09 87 65 44"}
    },
    {"IDVendeur", "Prénom", "Telephone"}
    ),
  Transformeletype= Table.TransformColumnTypes(Source,
{{"IDVendeur", Int64.Type}, {"Prénom", type text}, {"Telephone",
type text}})
in Transformeletype
```

© Editions ENI - All rights reserved