**Entrepôt de données**

**Rapport Final**

***Sébastien Quiquerez***

***Johan Steiner***

Table des matières

[1. Introduction 5](#_Toc472701853)

[2. Analyse de l’existant 5](#_Toc472701854)

[2.1. Fichier CSV délimité 5](#_Toc472701855)

[2.2. Fichier Microsoft Access Database 5](#_Toc472701856)

[2.2.1. Devises 6](#_Toc472701857)

[2.2.2. Vendeurs 6](#_Toc472701858)

[2.3. Fichier Microsoft Excel 6](#_Toc472701859)

[2.4. Fichier texte non délimité 6](#_Toc472701860)

[2.5. Fichier texte délimité 7](#_Toc472701861)

[2.6. Transmission d’informations sans fichier 7](#_Toc472701862)

[2.7. Structure et base de données 7](#_Toc472701863)

[3. Analyse des besoins 7](#_Toc472701864)

[3.1. Définition des contraintes sur les données 7](#_Toc472701865)

[3.2. Importation/transformation automatique des données 8](#_Toc472701866)

[3.3. Gestion des erreurs 8](#_Toc472701867)

[3.4. Structure de stockage unique 8](#_Toc472701868)

[3.5. Indicateurs 8](#_Toc472701869)

[3.6. Mise en place de hiérarchies 9](#_Toc472701870)

[3.7. Contrôle 9](#_Toc472701871)

[4. Cahier des charges 9](#_Toc472701872)

[4.1. Analyse des données 10](#_Toc472701873)

[4.2. Réalisation d’un modèle en étoile 10](#_Toc472701874)

[4.3. Définition des hiérarchies 10](#_Toc472701875)

[4.4. Installation et mise en place des outils nécessaires pour le projet 10](#_Toc472701876)

[4.5. Création des bases de données et tables 10](#_Toc472701877)

[4.6. Importation des données « brutes » dans la Staging Area 10](#_Toc472701878)

[4.7. Création du Master Data 10](#_Toc472701879)

[4.8. Création du processus de chargement des dimensions et tables de faits 11](#_Toc472701880)

[4.9. Création et mise en place du système d’audit 11](#_Toc472701881)

[4.10. Chargement des données dans les dimensions 11](#_Toc472701882)

[4.11. Chargement des données dans les tables de faits 11](#_Toc472701883)

[4.12. Génération d’un cube OLAP 11](#_Toc472701884)

[5. Modèle dimensionnel en étoile 12](#_Toc472701885)

[Explications des choix 12](#_Toc472701886)

[5.1.1. Tables de faits 12](#_Toc472701887)

[5.1.2. Mini dimension Commande 13](#_Toc472701888)

[5.1.3. Dimension Temps : Liaisons multiples 13](#_Toc472701889)

[5.1.4. Dimensions Client et Geographie : flocon 13](#_Toc472701890)

[5.2. Hiérarchies 14](#_Toc472701891)

[5.2.1. Hiérarchie du temps 14](#_Toc472701892)

[5.2.2. Hiérarchie des vendeurs 14](#_Toc472701893)

[5.2.3. Hiérarchie des produits 15](#_Toc472701894)

[5.2.4. Hiérarchie de la géographie 15](#_Toc472701895)

[6. Données : importation/transormation 15](#_Toc472701896)

[6.1. Staging 15](#_Toc472701897)

[6.1.1. Types de données 16](#_Toc472701898)

[6.1.2. Champs calculés et clés primaires 16](#_Toc472701899)

[6.2. Data Warehouse 16](#_Toc472701900)

[6.2.1. Master data 16](#_Toc472701901)

[6.2.2. Qualité des données 17](#_Toc472701902)

[6.2.3. Audit 17](#_Toc472701903)

[6.2.4. Importation des prix 18](#_Toc472701904)

[6.2.5. Problèmes lors du chargement des données 19](#_Toc472701905)

[6.2.6. Solution pour le chargement des données 19](#_Toc472701906)

[6.3. Chargement incrémental 19](#_Toc472701907)

[6.4. Cube 20](#_Toc472701908)

[7. Indicateurs 21](#_Toc472701909)

[7.1. Chiffre d’affaires total 21](#_Toc472701910)

[7.2. Chiffre d’affaires par année 21](#_Toc472701911)

[7.3. Chiffre d’affaire par devise 21](#_Toc472701912)

[7.4. Meilleur client jusqu’à aujourd’hui 22](#_Toc472701913)

[7.5. Trimestre générant le moins de revenus 22](#_Toc472701914)

[7.6. Nombre de jours de retard de livraison total par année 22](#_Toc472701915)

[7.7. Produit le plus vendu 23](#_Toc472701916)

[7.8. Produit générant le plus de marge 23](#_Toc472701917)

[7.9. Somme des ventes par département (région) 23](#_Toc472701918)

[7.10. Top 5 des vendeurs 24](#_Toc472701919)

[7.11. Bénéfice réel par année 24](#_Toc472701920)

[8. Conclusion 24](#_Toc472701921)

[9. Bibliographie 25](#_Toc472701922)

# Introduction

Mediazon est spécialisée dans la vente de produits multimédias. L’entreprise nous a mandaté pour mettre en place la structure de son Data Warehouse qui stockera de grandes quantités de données issues des commandes de ses clients.

Les données source sont donc sous une forme « brute », provenant de différentes sources, avec un certain nombre d’erreurs et il est nécessaire de mettre en place un système permettant d’obtenir des données d’une certaine qualité afin de permettre le développement d’un système décisionnel bien conçu.

Le développement du système de test a été réalisé depuis une machine virtuelle Microsoft Server 2012. Le logiciel Microsoft Visual Studio ainsi que Microsoft SQL Server Management Studio nous ont permis de réaliser la plus grande partie de notre travail.

# Analyse de l’existant

Au départ, les données sont dispersées entre plusieurs sources de données de différents types. A notre disposition, nous avons les types de sources suivants :

* Fichier CSV délimité
* Fichier Microsoft Excel
* Fichier Microsoft Access Database
* Fichier texte non délimités
* Fichier texte délimité
* Transmission d’informations sans fichier

Les informations contenues dans les différents fichiers sont décrites ci-dessous.

## Fichier CSV délimité

Dans ce fichier, les différentes villes de France sont présentes, avec leur NPA respectif et le département dans lequel elles se situent. Chaque ville est identifiée par un ID unique.

## Fichier Microsoft Access Database

Ce fichier contient 2 catégories de données distinctes.

### Devises

Mediazon livre ses produits dans plusieurs régions du monde, ce qui signifie qu’elle doit adapter la devise de paiement en fonction de ces régions. Elle possède donc la liste des devises présentes dans le monde avec, pour chacune d’elles, un code à trois chiffres uniques.

### Vendeurs

L’entreprise possède également une liste de ses vendeurs avec pour chacun d’eux son nom et prénom, la division et le groupe dans laquelle il travaille, ainsi que son manager respectif.

En plus de cela, des informations plus personnelles telles que le sexe de la personne, son quota, bonus ou commission sont renseignées. Les vendeurs sont aussi identifiés à l’aide d’un ID unique.

## Fichier Microsoft Excel

Un fichier Microsoft Excel composé de la liste des différents produits que l’entreprise vend nous a été mis à disposition.

On peut y trouver des informations sur le produit comme par exemple son titre, son auteur, son prix (catalogue et de base) ou encore les catégories et sous-catégories auxquelles il appartient. La référence du produit permet de l’identifier car elle est unique.

## Fichier texte non délimité

Le fichier texte non délimité ne contient pas de délimiteur (virgule, point-virgule, etc…) mais chaque « champ » a un nombre de caractères défini. Si le contenu de ce champ est plus petit que le nombre défini, les caractères disponibles restants sont comblés par des espaces. Cela crée des « pseudo-colonnes ».

Les fichiers textes ne contiennent pas titres de colonnes, néanmoins nous avons eu les informations sur la signification des valeurs/textes présents dans chaque « colonne ».

Ce fichier contient donc les informations globales des commandes passées par nos clients, leurs « en-têtes ». Le détail de celles-ci se trouve dans un autre fichier, celui-ci nous renseigne sur des éléments plus généraux tels que le numéro de la commande, les différentes dates (commande, livraison prévue, expédition si déjà expédié), la devise, les informations personnelles du client concerné, l’expéditeur de la commande ou encore le code du vendeur ayant réalisé la vente.

Parmi tous ces éléments, des identifiants uniques sont présents, comme le numéro de la commande, un ID pour le client, le code du vendeur ou encore celui de la devise.

## Fichier texte délimité

Ici, ce sont les lignes de nos commandes qui sont présentes. Logiquement, ce fichier contient plus d’enregistrements que celui des en-têtes étant donné qu’une commande contient souvent plusieurs lignes. On s’intéresse donc plus au produit vendu qu’à la commande globale.

On retrouve donc diverses informations sur le produit (titre, auteur, prix), la quantité commandée pour chaque produit et la remise sur le prix unitaire de celui-ci. La marge unitaire est également indiquée.

Chaque produit est identifiable par sa référence qui est unique.

## Transmission d’informations sans fichier

D’autres informations importantes nous ont été transmises mais elles ne sont pas inscrites dans un fichier depuis lequel nous pourrions directement les reprendre. Il s’agit par exemple des informations sur les expéditeurs, par exemple la liste des différentes compagnies d’expédition avec lesquelles l’entreprise travaille, leurs coûts de base et unitaires pour l’expédition des produits.

## Structure et base de données

Au début du projet, la base de données n’est pas encore en place. Les tables n’existent donc pas encore et l’importation des données n’a pas encore été réalisée. Nous partons donc d’une base « vide » et brut (en ce qui concerne les données) à ce niveau.

# Analyse des besoins

Plusieurs besoins ressortent au début du projet.

## Définition des contraintes sur les données

Avant la définition des contraintes, il faudra tout d’abord comprendre le sens des données. Certains fichiers ne contiennent donc pas d’en-tête et ne sont pas forcément délimités. Il faudra savoir ce que chaque colonne de nos données représente et pouvoir créer des « colonnes » correctes.

A l’heure actuelle, les données se trouvent dans un format « brut », ce qui signifie qu’elles sont sous forme de simple texte, sans aucune contrainte de taille, sans interdiction d’une valeur nulle, etc…

Il est donc nécessaire de définir des contraintes sur celles-ci. Il va notamment falloir définir quelle donnée devra obligatoirement ou non obligatoirement être renseignée pour chaque enregistrement. Certaines données nécessiteront des contrôles (CHECK) sur la valeur entrée afin qu’elle respecte par exemple une liste de valeurs autorisées.

## Importation/transformation automatique des données

A partir des données brutes recueillies dans les différents documents cités ci-dessus, nous devons effectuer divers traitements sur les données afin qu’elles respectent les contraintes définies précédemment et qu’elles soient converties dans le bon format (format *décimal* pour des prix par exemple).

Ces traitements devront être automatisés afin qu’à chaque chargement de nouvelles données ou modification de données existantes, les données ressortent dans le même format que celui souhaité et défini. Il ne devrait donc pas y avoir de génération d’erreur de la part du système qui sera mis en place.

## Gestion des erreurs

Il est certain que parmi les données à transformer, certaines généreront des erreurs. Par exemple, s’il y a une liaison à faire avec une autre donnée mais que celle-ci n’est pas trouvée, cela stoppera le processus de traitement.

Il va donc falloir mettre en place une gestion d’erreur appropriée pour que, malgré certaines données erronées, l’ensemble des enregistrements que l’on possède soit traité et, si possible, validé. On pourrait par exemple imaginer remplacer les valeurs erronées par une valeur définie comme « standard pour les erreurs ».

## Structure de stockage unique

Actuellement, nos données sont stockées dans plusieurs types de fichiers. La mise en commun des informations est donc actuellement impossible. En effet, il serait compliqué d’unir les données un document Access avec un document Excel (ou du moins, pas « confortable » pour les combiner).

Il est donc nécessaire que toutes nos données puissent être stockées à un seul et même endroit où il est possible de les utiliser ensemble afin de pouvoir mettre en place des indicateurs par exemple. Dans cette même structure, on pourra les extraire avant leur transformation, cela permettra de donner à nos processus de traitements une source unique où trouver les données qui provenaient auparavant des divers types de fichiers à notre disposition.

## Indicateurs

Une fois que les données auront été transformées et importées dans leur version finale, elles pourront être utilisées pour créer des indicateurs.

Ceux-ci devront être définis de manière cohérente afin qu’ils soient pertinents. Le but ne sera pas d’en avoir en qualité mais d’avoir des indicateurs de qualité.

## Mise en place de hiérarchies

Pour plusieurs de nos données, il serait possible de créer des hiérarchies qui permettraient de créer des indicateurs de différentes granularités (unité de temps : jour / mois / année).

## Contrôle

Lors du chargement des données, il sera nécessaire de mettre en place des éléments permettant de contrôler le processus : est-ce que celui-ci s’est bien déroulé ? Y a-t-il eu des éléments qui ont créé des problèmes ?

# Cahier des charges

Un cahier des charges pour le projet a été défini.

|  |  |
| --- | --- |
| **N° tâche** | **Tâche** |
| **1** | Analyse des données |
| **2** | Réalisation d’un modèle en étoile |
| **3** | Définition des hiérarchies |
| **4** | Installation et mise en place des outils nécessaires pour le projet |
| **5** | Création des bases de données et tables |
| **6** | Importation des données « brutes » dans la Staging Area |
| **7** | Création du Master Data |
| **8** | Création du processus de chargement des dimensions et tables de faits |
| **9** | Création et mise en place du système d’audit |
| **10** | Chargement des données dans les dimensions du Data Warehouse |
| **11** | Chargement des données dans les tables de faits du Data Warehouse |
| **12** | Génération d’un cube OLAP |

## Analyse des données

Les données doivent tout d’abord être analysées afin de définir lesquelles d’entre elles seront utiles dans notre système. Il faut également déterminer comment celles-ci seront transformées (quel type ? quelles contraintes ? etc.).

## Réalisation d’un modèle en étoile

Un modèle en étoile composé des différentes dimensions et tables de faits qui seront présentes dans notre système doit être mis en place pour avoir une idée plus claire de celui-ci.

## Définition des hiérarchies

Il est également nécessaire de définir les différentes hiérarchies des dimensions pour que les éléments utiles à leur mise en place puissent être créés.

## Installation et mise en place des outils nécessaires pour le projet

Avant de pouvoir débuter le développement du projet, il faudra installer et mettre en place les différents composants (logiciels, environnement, etc.) utiles à la création de notre système.

## Création des bases de données et tables

Une fois l’environnement préparé, nous devons créer les bases de données et les tables qui les composent. Une base de données, la Staging Area, sert de zone intermédiaire (entre les fichiers sources et la destination) et une autre, de base de données finale contenant les données transformées.

## Importation des données « brutes » dans la Staging Area

Lorsque les bases de données sont créées, il faut importer les données des fichiers sources dans la Staging Area.

## Création du Master Data

Une gestion des données de référence (Master Data) doit également être mise en place pour garantir l’unicité, l’intégrité ou encore la fiabilité de certaines données.

## Création du processus de chargement des dimensions et tables de faits

Il faut ensuite mettre en place un processus automatique de chargement des données dans nos dimensions et tables de faits du Data Warehouse. Cela concerne donc la transformation des données.

## Création et mise en place du système d’audit

A des fins de contrôle sur le déroulement du processus précédemment créé, nous devons mettre en place un système d’audit. Celui-ci doit être intégré au processus de chargement des données dans le Data Warehouse et permettre de garder une trace de celui-ci, de son déroulement, des éventuelles erreurs rencontrées, etc.

## Chargement des données dans les dimensions

Quand la structure de stockage, les processus de chargement des données et le système d’audit sont mis en place, nous devons importer nos données dans les dimensions de notre Data Warehouse et vérifier le bon déroulement du processus de chargement. Si des erreurs surviennent, il est essentiel de les corriger.

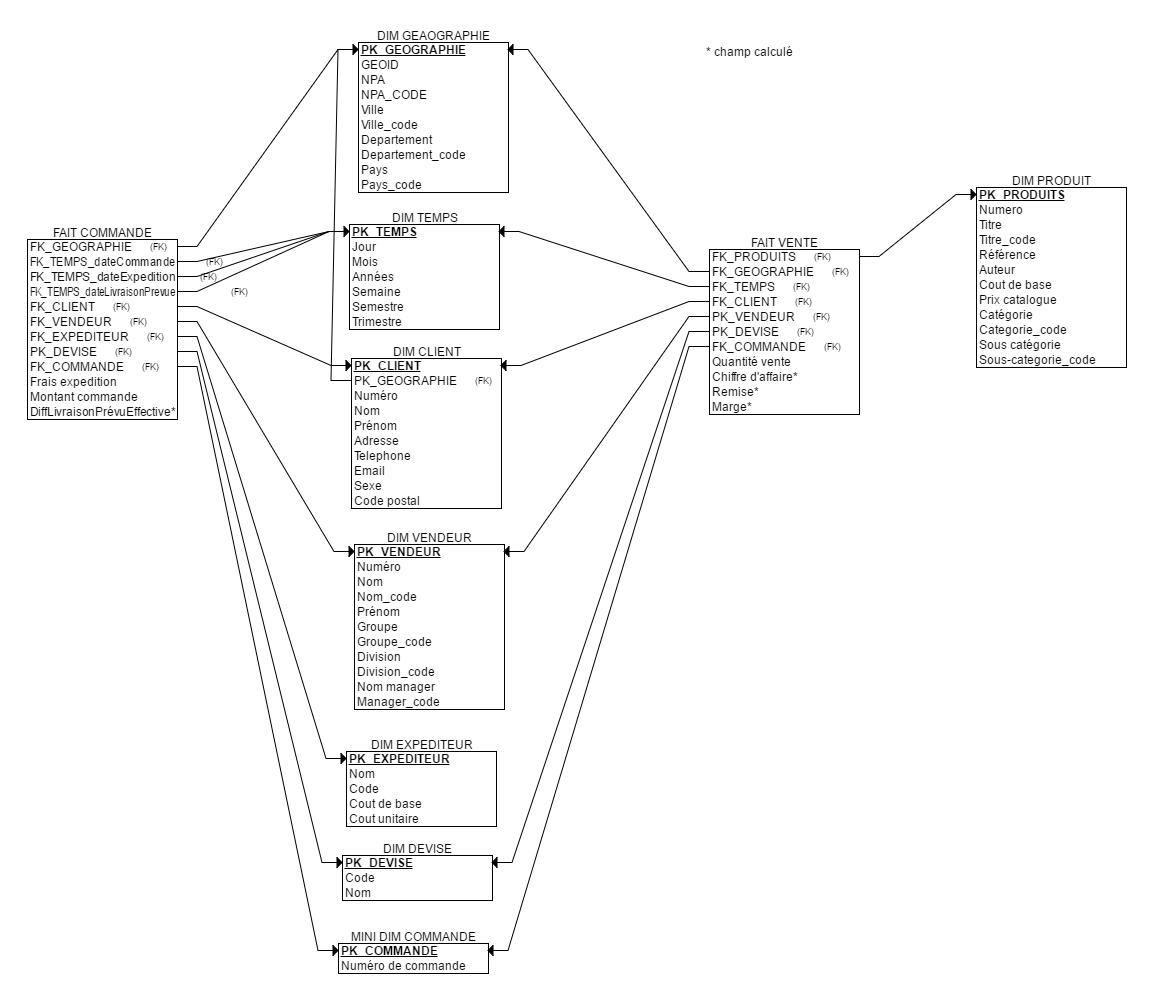
## Chargement des données dans les tables de faits

De même que pour le chargement des données dans les dimensions, nous devons charger les enregistrements dans nos tables de faits et corriger les éventuelles erreurs.

## Génération d’un cube OLAP

Une fois le chargement des données effectué avec succès, nous créons une nouvelle structure, un cube OLAP, qui permettra une gestion simplifiée des indicateurs que nous avons définis.

# Modèle dimensionnel en étoile



Explications des choix

### Tables de faits

Notre schéma contient 2 tables de faits : la table **Commande** et la table **Vente**.

La table *Vente* contient les informations propres aux produits et plutôt à une ligne de la commande (En quelle quantité un produit a-t-il été acheté ? De quel pourcentage de remise a-t-on bénéficié sur chaque produit ? Quelle marge a été réalisée sur la vente ? Etc…).

La table *Commande* est plus orientée sur la commande en générale, par exemple pour connaître les dates de commande, livraison et expédition, l’expéditeur ainsi que ses frais ou encore le client qui a passé la commande.

### Mini dimension Commande

Avant la création de cette dimension, il était impossible de lier une commande aux ventes. On ne pouvait par exemple pas connaître le prix total de la commande.

La mini dimension Commande permet de créer cette liaison : elle contient le numéro de chaque commande et dans chaque table de faits, une clé secondaire référence cette table afin qu’il soit possible de faire correspondre les informations des deux tables de faits.

Par exemple, si l’on souhaite connaitre le montant total des produits d’une commande (laissons de côté les frais d’expédition pour cet exemple), il suffit de multiplier les prix d’un produit par sa quantité et de répéter cette opération pour chaque produit de la commande. Grâce à notre mini dimension Commande, nous pouvons lier le numéro de commande de la table de faits *Commande* à la table de faits *Vente* et donc savoir quel produit appartient à quelle commande.

### Dimension Temps : Liaisons multiples

Nous avons créé 3 liaisons entre la dimension Temps et la table de faits Commande, car dans le cas, il est mentionné 3 types de dates distinctes :

* La date de commande
* La date de livraison prévue
* La date d’expédition

### Dimensions Client et Geographie : flocon

Un flocon a été mis en place entre les tables Client et Geographie car nous avons jugé intéressant de connaitre la provenance de nos clients sans qu’on doive forcément se lier à une vente. Cela permettra la mise en place d’indicateurs supplémentaires.

## Hiérarchies

### Hiérarchie du temps

**Année**

**Année**

**Semaine**

**Mois**

**Semestre**

**Jour**

**Jour**

**Trimestre**

**Mois**

**Jour**

### Hiérarchie des vendeurs

**Groupe**

**Division**

**Manager**

**Vendeur**

### Hiérarchie des produits

**Catégorie**

**Sous-catégorie**

**Article**

### Hiérarchie de la géographie

**Pays**

**Département**

**Ville**

# Données : importation/transormation

L’importation des données dans notre Data Warehouse a été réalisée à l’aide de la méthode ETL (Extract, Transform, Load).

## Staging

La zone Staging représente la zone contenant les données qui n’ont pas encore été transformées. Nous extrayons les données depuis nos différentes sources (fichiers Excel, fichiers Access, fichiers texte, etc…), les plaçons dans notre Staging area, à la suite de quoi nous les transformons (conversion de type, modification de la longueur des champs, etc…) pour qu’elles correspondent à ce qu’attendent les tables présentes dans notre Data Warehouse et les chargeons dans celui-ci.

### Types de données

L’importation des données dans la Staging a été simplifiée au niveau des types de données. En effet, nous n’avons à ce moment-là pas encore procédé à une transformation des différents types, mais avons simplement importé toutes les données sous forme de chaînes de caractères.

### Champs calculés et clés primaires

Nos tables de faits contiendront des champs calculés (montant total d’une commande par exemple) ainsi que des clés primaires pour identifier les enregistrements. Ces deux éléments n’apparaissent pas encore dans la Staging, ils seront créés par la suite.

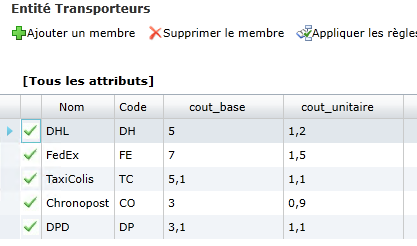
## Data Warehouse

### Master data

Les données de référence (Master Data) dans une entreprise sont partagées entre les différents systèmes. Elles représentent donc un objet métier de base, partagées entre toutes les applications de l'entreprise.

Ces données de référence constituent également une source de vérité entre toutes les applications qui les entourent. Elles permettent de garder des données fiables, maintenir l’intégrité de celles-ci ainsi que leur unicité.

Prenons un exemple qui s'applique à notre cas :

On souhaite modifier un expéditeur dans l’un des nombreux systèmes de notre entreprise. On aimerait donc que cette modification se répercute sur tous les autres systèmes afin de n'avoir toujours qu'un et un seul expéditeur avec toutes ses données correctes et à jour. On veut donc que l’expéditeur soit toujours le même à n'importe quel moment et n'importe où dans les systèmes de notre entreprise.

Dans notre flux qui permet d'alimenter notre dimension des expéditeurs, nous reprenons tous les expéditeurs qui sont dans nos commandes. Avant de les insérer dans notre dimension, on interroge notre Master Data afin de contrôler qu'il existe bien dans notre système. Si cela est le cas, on l'insère dans notre dimension. Le cas échéant, on place le nom de l’expéditeur inconnu dans un fichier de rejet afin de le traiter manuellement.

*Master Data pour nos expéditeurs*

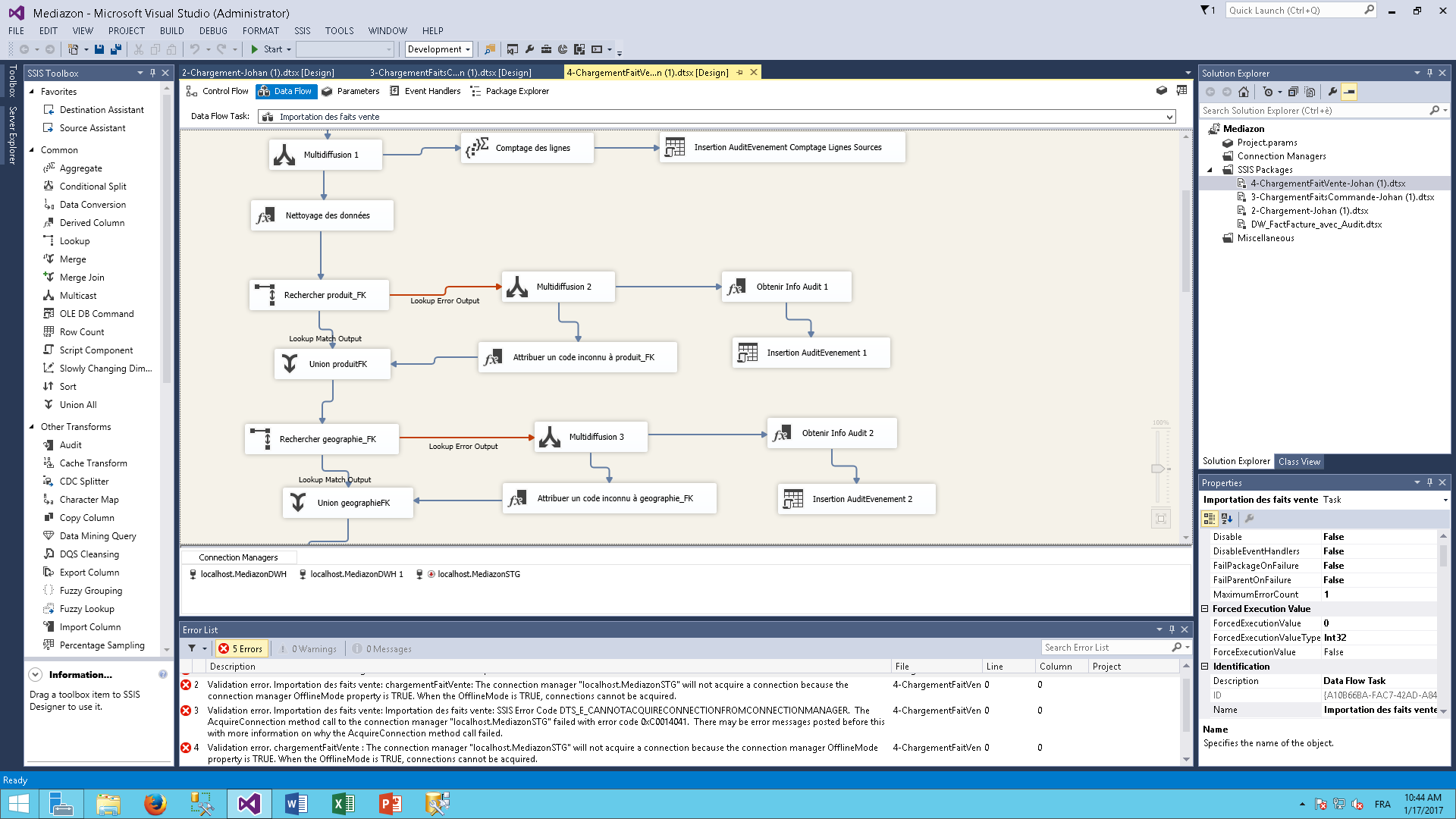
Si l'on traite un expéditeur rejeté alors qu'il existe bien dans notre Master Data (le motif du rejet pourrait être une simple faute de frappe), on ne le rajoutera pas dans la dimension ni dans le Master Data. Au contraire, s’il s'agit d'un nouvel expéditeur, on l'ajoutera par la suite dans le Master Data et dans la dimension. Cela aura pour effet de ne plus rejeter l’expéditeur lors de la recherche dans le Master Data durant le prochain chargement des données.

### Qualité des données

Il a fallu que nous effectuions un choix au niveau de la qualité des données importées dans nos tables de faits. Nous avions 2 solutions :

1. **Eliminer les enregistrements incorrects, générant une erreur**
2. **Inclure les enregistrements incorrects en corrigeant leurs erreurs**

Nous avons opté pour la seconde solution. Nous souhaitons que nos tables de faits contiennent la totalité des enregistrements et non seulement ceux qui sont corrects. Cela permettra d’avoir des indicateurs plus corrects. En effet, supprimer une commande parce qu’elle n’est pas correcte fausserait également un indicateur comme « nombre de commandes passées en 2016 ».

Chaque table contient un enregistrement « Inconnu » (clé=0 / nom= Inconnu » / etc…). Lors de l’import des données dans une table de faits, une recherche est effectuée sur les clés secondaires contenues dans cette table pour faire la liaison avec le bon enregistrement de la dimension correspondante. Il arrive cependant que le champ contienne une valeur non existante dans celle-ci. Cet enregistrement permet justement de régler ce problème : au lieu de générer une erreur, les enregistrements n’ayant pas trouvé de liaison sont redirigés vers un traitement qui leur assigne cette valeur au niveau de leur clé secondaire. La suite du processus peut ensuite se dérouler sans problème. Ce traitement est également utilisé pour les flocons réalisés sur nos dimensions.

*Assignation de la valeur* ***Inconnu***

### Audit

Nous avons mis en place un système d’audit au sein de notre système, à des fins de traçabilité et contrôle (indication des erreurs).

3 tables composent ce système :

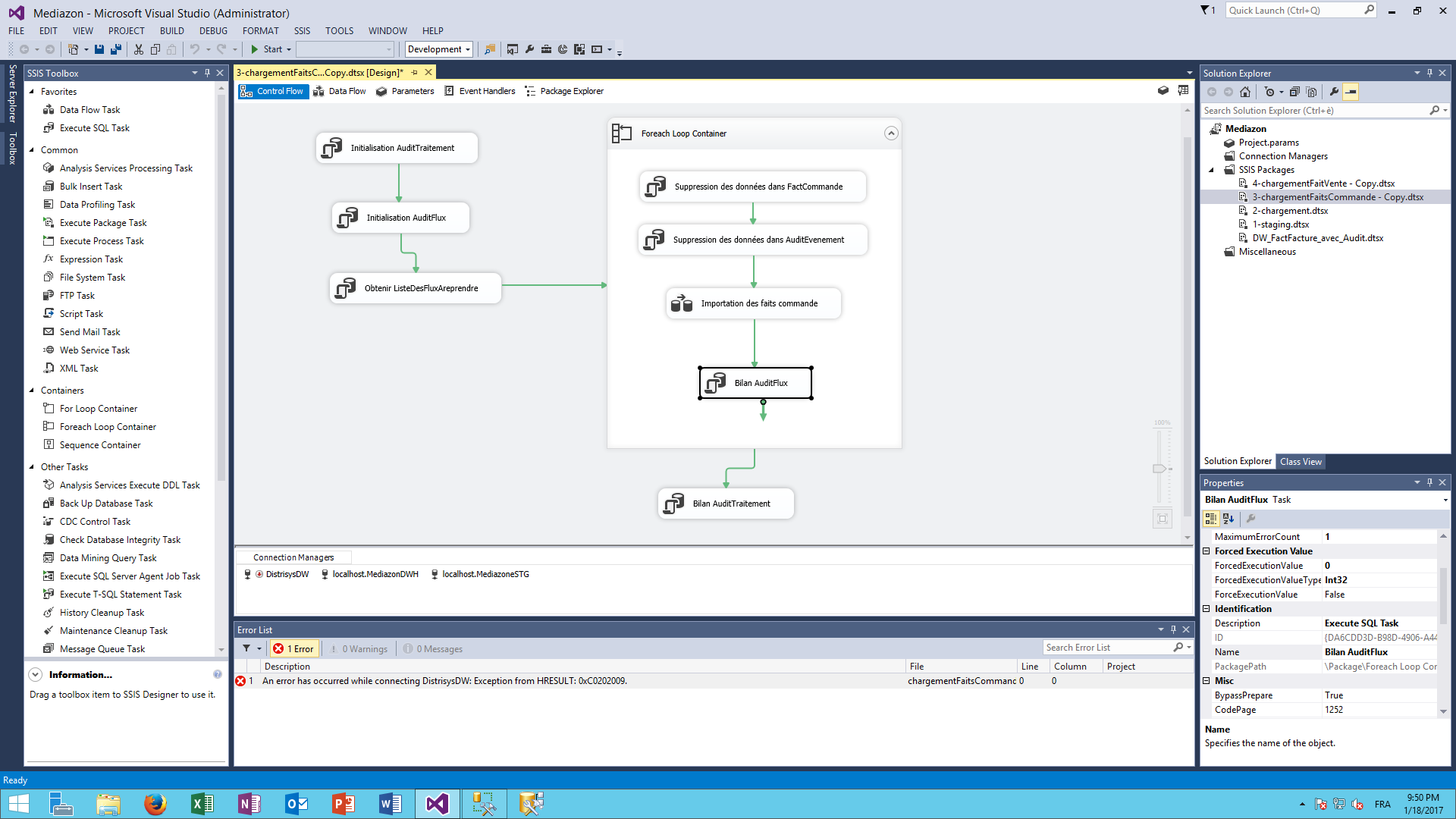
* Table AuditTraitement
* Table AuditFlux
* Table AuditEvenement

La table **AuditTraitement** permet un compte rendu global du résultat final des exécutions de notre flux (succès ? nombre d’erreurs ? début et fin du traitement ? etc.). Elle retourne un seul enregistrement, même si le flux a dû être exécuté à plusieurs reprises.

La table **AuditFlux** fait le bilan de l’exécution d’un flux particulier. Elle contient plus d’informations sur le flux que l’enregistrement lié à ce flux, présent dans la table AuditTraitement (nombre d’exécution du flux ? etc.). Les flux qui ont généré des erreurs techniques peuvent donc être repris et exécutés à nouveau par le système.

La table **AuditEvenement** recense, quant à elle, chaque ligne générant une erreur ou un avertissement. Chaque erreur/avertissement donne lieu à l’insertion d’un enregistrement dans cette table. Elle est liée à la table AuditFlux, qui rassemble les erreurs pour un flux donné.

Le schéma d’exécution d’un processus avec un système d’audit est le suivant :



### Importation des prix

L’un de nos choix d’importation concerne l’importation des prix et calcul du montant de la commande : du fait que les commandes sont payées en différentes devises, nous avions en effet le choix entre une transformation en une devise unique avant l’importation dans notre Data Warehouse ou une simple importation dans la devise de base.

Nous avons choisi de garder la devise de base lors de l’importation dans nos tables de fait. Ce choix s’explique du fait que nous avons pris en considération les variations du taux de change des devises. Voici un exemple :

Admettons que nous possédons un livre qui coûte 100€ avec un taux de change de 1.2 entre l’euro et le franc suisse (CHF). Admettons encore que quelques mois plus tard, ce taux a diminué à 1.1. Si nous avions converti le prix de notre produit en CHF à l’importation, celui-ci aurait valu 120 CHF et, dans le cas où nous aimerions savoir son prix actuel en magasin en euro, il vaudrait théoriquement 109€ alors qu’en magasin, il est certainement toujours vendu à 100€, soit actuellement 110 CHF.

Nous avons donc préféré opter pour un prix « à jour », qui sera, si besoin, converti en temps réel avec le taux de change actuel pour sa monnaie.

### Problèmes lors du chargement des données

Durant le chargement des données, nous avons rencontré des problèmes qui ont engendré des erreurs dans notre système.

Le chargement depuis les deux fichiers sources (fichiers textes) dans notre Staging Area n’a engendré aucun problème. Toutes les données ont été importées avec succès dans nos tables (une pour les en-têtes des commandes et l’autre pour les lignes composant ces commandes). Cependant, lors du processus de transformation et chargement dans le Data Warehouse, nous avons rencontré des problèmes qui ont généré des erreurs. Au total, sur les 540'000 lignes des commandes (chiffre approximatif), environ 20'000 n’ont pas pu être chargées.

### Solution pour le chargement des données

Nous avons donc recherché la cause des erreurs et avons découvert un oubli dans notre processus de chargement.

Les données concernant les commandes de nos clients comportaient 3 dates :

* Date de commande
* Date de livraison prévue
* Date d’expédition

Lors de la création de la requête SQL reprenant les données utiles présentes dans notre Staging Area pour les intégrer à notre Data Warehouse, nous avions pris en compte le fait que la date d’expédition n’était pas toujours renseignée. Nous gérions donc cette problématique en remplaçant la valeur *null* par une valeur inconnue qui respectait néanmoins le type de données et permettait les calculs sur les dates.

Cependant, nous n’avions pas géré cette problématique pour les dates de livraison prévue. Ce qui a généré une erreur lors de la réalisation des calculs sur les dates. Certains enregistrements n’ont donc pas pu terminer le processus et être ajoutés à nos tables de faits.

L’erreur a été corrigée par la suite. Le processus a été relancé et l’ensemble des enregistrements a pu être chargé dans nos tables de faits.

## Chargement incrémental

Comme nous avons mis en place un système d'audit dans notre solution avec une reprise automatique des flux qui ne sont pas passés correctement, nous avons également pu gérer le fait de pouvoir réaliser des chargements incrémentaux dans nos tables de faits afin de ne reprendre que les nouvelles données qui arrivent dans notre entrepôt et ceci pour éviter de recharger tous les faits entièrement (ce qui prendrait trop de temps). Cela se déroule de la manière suivante :

Nous exécutons les flux de chargement de nos tables de faits. À la fin de l'exécution, un bilan de cette exécution est fait et est enregistrée dans une table qui recense tous les audits. Si le flux s'est déroulé correctement, donc sans erreur ni avertissement, il n'aura pas besoin d'être exécuté à nouveau la prochaine fois et aura donc la valeur ***N*** dans la colonne ***FluxAreprendre***. Au contraire, si des erreurs ou avertissements surviennent durant l'exécution du flux, la valeur enregistrée à la fin sera ***O*** afin d'indiquer que le flux ne s'est pas déroulé correctement et qu'il devra donc être repris lors de la prochaine exécution. Si le flux doit être repris, tous les faits déjà importés dans la table seront supprimés en fonction du numéro du flux qui les a ajoutés (le flux à reprendre). Si le flux ne doit pas être repris, aucune ligne de la table de faits ne sera supprimée et il n'y aura que les nouvelles lignes qui viendront s'y ajouter.

Dans notre cas, nous avons dû recharger complètement nos tables de fait car nous avons eu des avertissements lors des chargements précédents. Cela provient du fait que les lignes qui n'ont pas trouvé de correspondance dans nos ***lookup*** sont considérées comme des avertissements et donc, qui dit avertissement dit flux à reprendre.

## Cube

Une fois l’importation des données dans notre Data Warehouse réalisée, nous avons pu passer à la génération de notre cube.

Dans notre système, nous avons donc créé deux cubes. Ces deux cubes correspondent aux deux tables de faits que nous avons créé dans notre Data Warehouse. A chacun des cubes sont rattachées les dimensions liées à la table de faits correspondant au cube. Pour chaque dimension, nous avons ajouté les attributs les plus pertinents pour l'utilisateur afin qu'il puisse effectuer des agrégations avec des éléments qu'il comprend (nous ne lui fournissons pas, par exemple, de clés primaires ou éléments réservés au développeur). Nous avons par exemple ajouté les attributs *nom* et *prénom* de la dimension *Personne*.

Une fois nos tables de faits et nos dimensions prises en compte pour nos cubes, nous avons ajouté toutes les hiérarchies que nous avions prévues préalablement (page 14 du document).

Nous avons également ajouté des mesures à nos cubes :

* Nombre de commandes
* Nombre de ventes

On peut constater que dans nos cubes des Commandes ou des Ventes, on retrouve toutes les valeurs calculées que nous avions créées précédemment lors du chargement dans nos tables de faits. La différence en nombre de jours entre la date de livraison prévue et la date d’expédition effective en est un très bon exemple.

Dès que nos cubes ont été correctement paramétrées avec les bonnes dimensions, hiérarchies mesures ainsi que les bons attributs, nous l'avons déployé. Les cubes nous servent à faire des analyses sur les données qui se trouvent dans notre entrepôt de données. On peut donc créer les différents indicateurs de performance (ou KPI) qui nous semblent pertinents afin d'avoir un bon outil d'aide à la décision.

# Indicateurs

Pour la réalisation de nos indicateurs, nous avons utilisé le logiciel Power BI.

## Chiffre d’affaires total

Le premier indicateur défini calcule la somme des montants de toutes les commandes de nos clients. Ces montants n’incluent pas les frais d’expédition, mais uniquement les sommes par rapport aux ventes de produits.

Ce premier indicateur est assez global et permet donc d’obtenir le chiffre d’affaires généré par l’ensemble des commandes présentes dans le système.

## Chiffre d’affaires par année

Le second indicateur défini affiche la somme des montants des commandes en fonction de l’année de la commande.

Celui-ci diffère donc du premier, du fait que le facteur année est pris en compte pour séparer les différentes sommes des commandes. On a donc une granularité différente.

La somme des montants de toutes les années nous donne donc le même résultat que le premier indicateur.

## Chiffre d’affaire par devise

La somme des montants des commandes a également été séparée en fonction des différentes devises de paiement. A l’heure actuelle, 4 devises différentes renvoient un résultat.

A nouveau, la somme des montants de toutes les devises nous renvoie au même résultat que le premier indicateur.

## Meilleur client jusqu’à aujourd’hui

Nous avons défini un indicateur affichant le meilleur client présent dans notre Data Warehouse avec les revenus qu’il a déjà générés pour notre entreprise. Cela permettrait de mettre en place une fidélisation du meilleur client avec éventuellement une récompense pour celui-ci, e qui renforcerait la relation avec celui-ci.

## Trimestre générant le moins de revenus

Les périodes de l’année ne comptabilisent pas toutes le même montant de chiffre d’affaires. En fonction de la période, le nombre de commandes et donc le chiffre d’affaires peuvent varier. Prenons par exemple la période de Noël : en 1 semaine, certainement que le chiffre d’affaires sera plus important que, par exemple, durant les 2 premières semaines du mois de septembre.

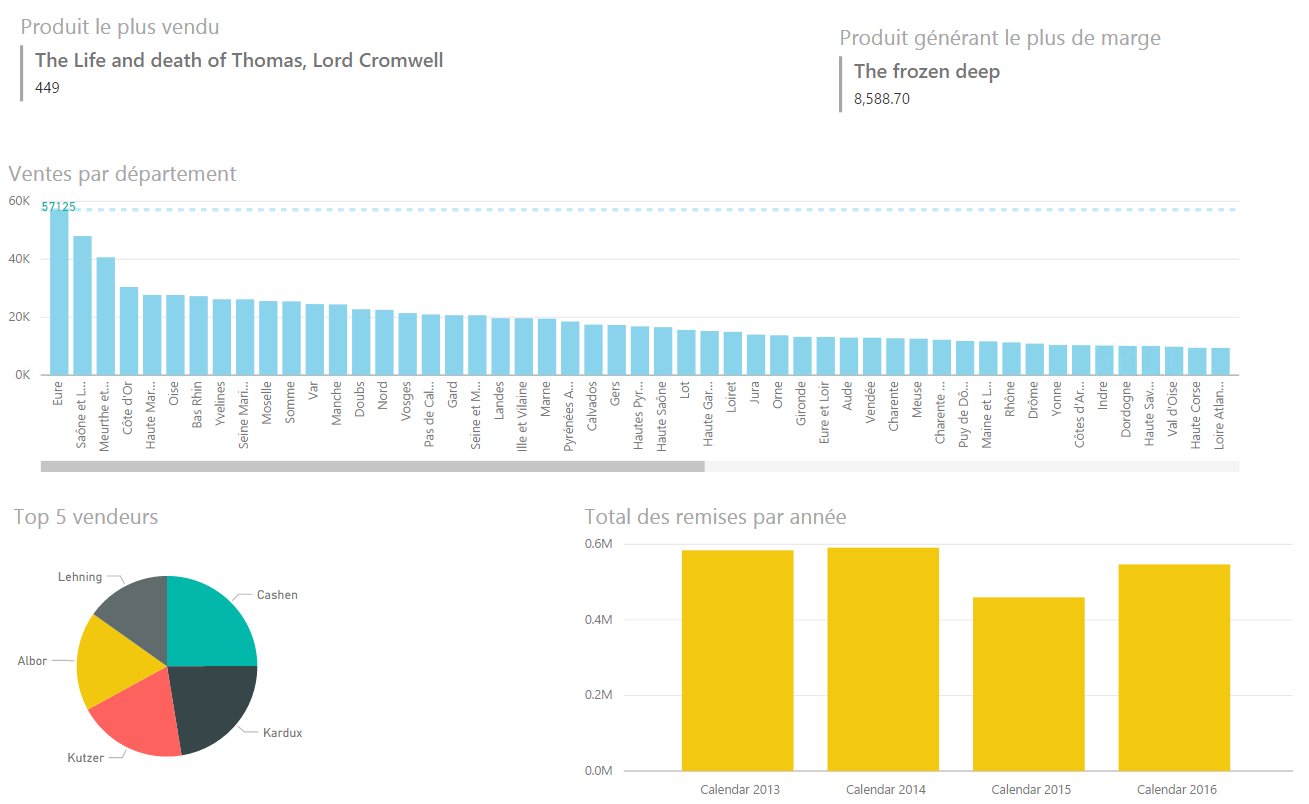
Grâce à cet indicateur, nous pourrions définir une stratégie précise. Par exemple, durant le trimestre générant le moins de chiffre d’affaires, le budget marketing pour nos produits pourraient être diminué au profit d’un trimestre générant plus de chiffre d’affaires.

## Nombre de jours de retard de livraison total par année

Afin d’évaluer notre capacité à respecter les dates de livraison approximatives données (date de livraison prévue), nous avons créé un indicateur calculant le nombre de jour total, par année, de retard par rapport à nos prévisions aux clients.

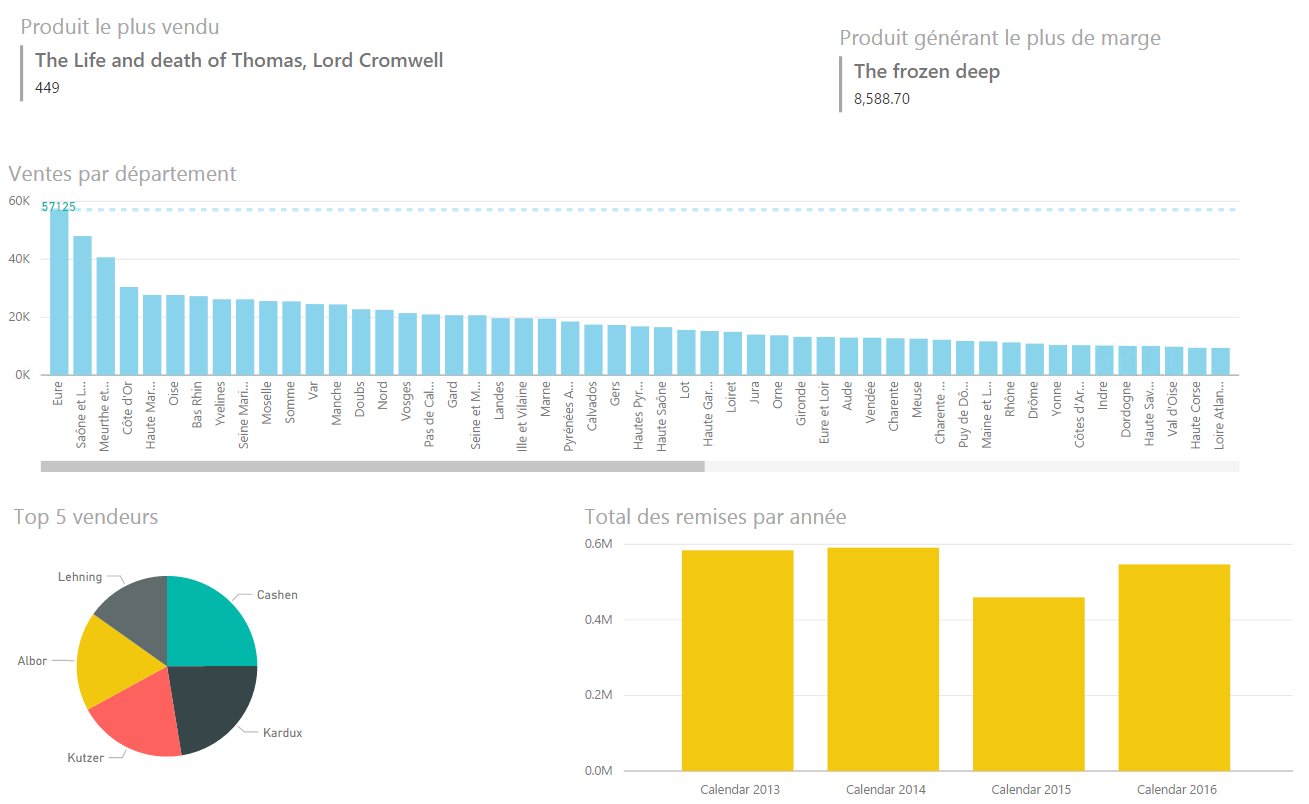
Par exemple, si la date de livraison prévue est du 15 février 2016 et que la date d’expédition réelle est le 17 février 2016, 2 jours de retard seront comptabilisés.

## Produit le plus vendu

Nous avons créé un indicateur permettant de faire ressortir le produit qui rencontre le plus de succès en termes de ventes, donc le produit le plus vendu.

Grâce à cela, nous pourrions mettre en avant ce produit afin de faire encore grandir davantage l’intérêt des clients pour celui-ci.

## Produit générant le plus de marge

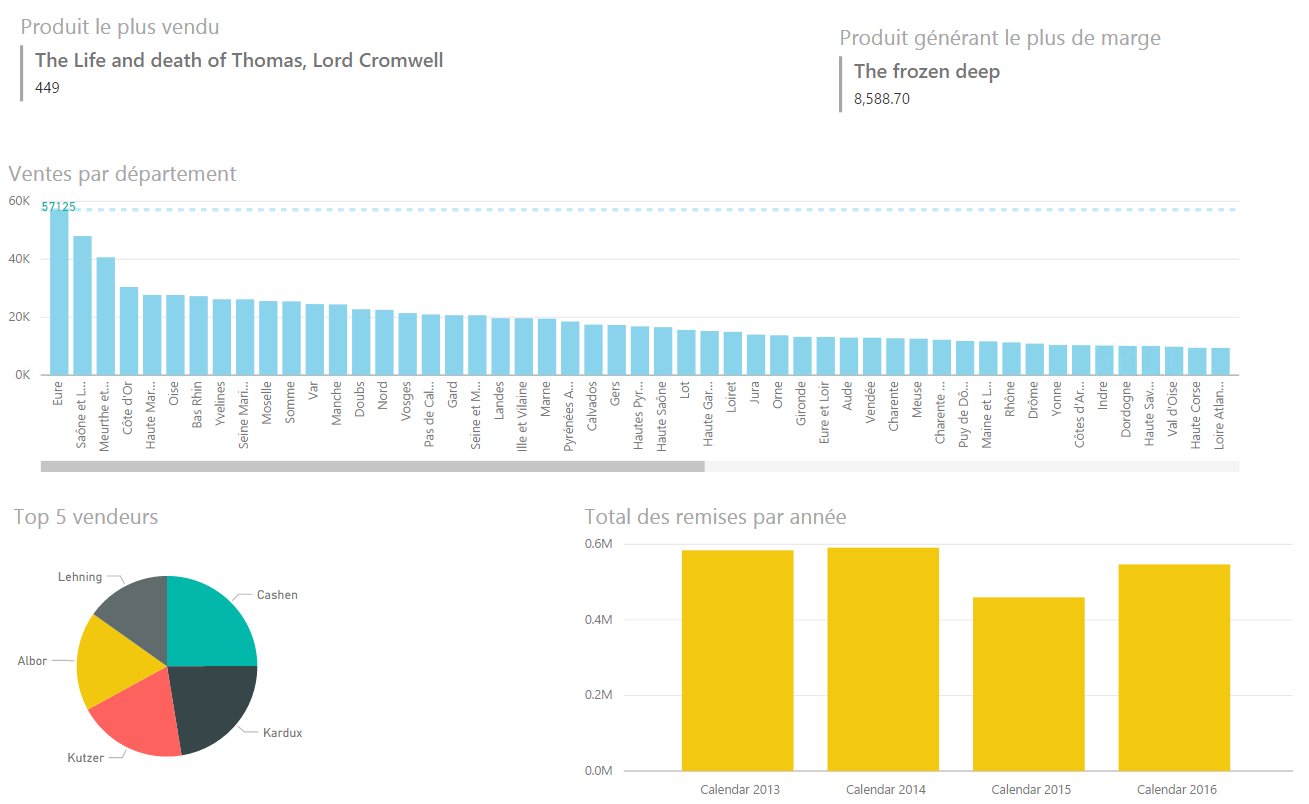
De plus, nous souhaitions connaître le produit qui a rapporté le plus de marge. Bien que certains produits se vendent mieux que d’autres, le produit le plus vendu n’est pas forcément le même que celui qui génère la plus grande marge, ce qui n’est justement pas le cas dans notre situation.

Il est donc important de connaître le produit sur lequel le bénéfice réel (chiffre d’affaire total – coûts totaux) est le plus élevé.

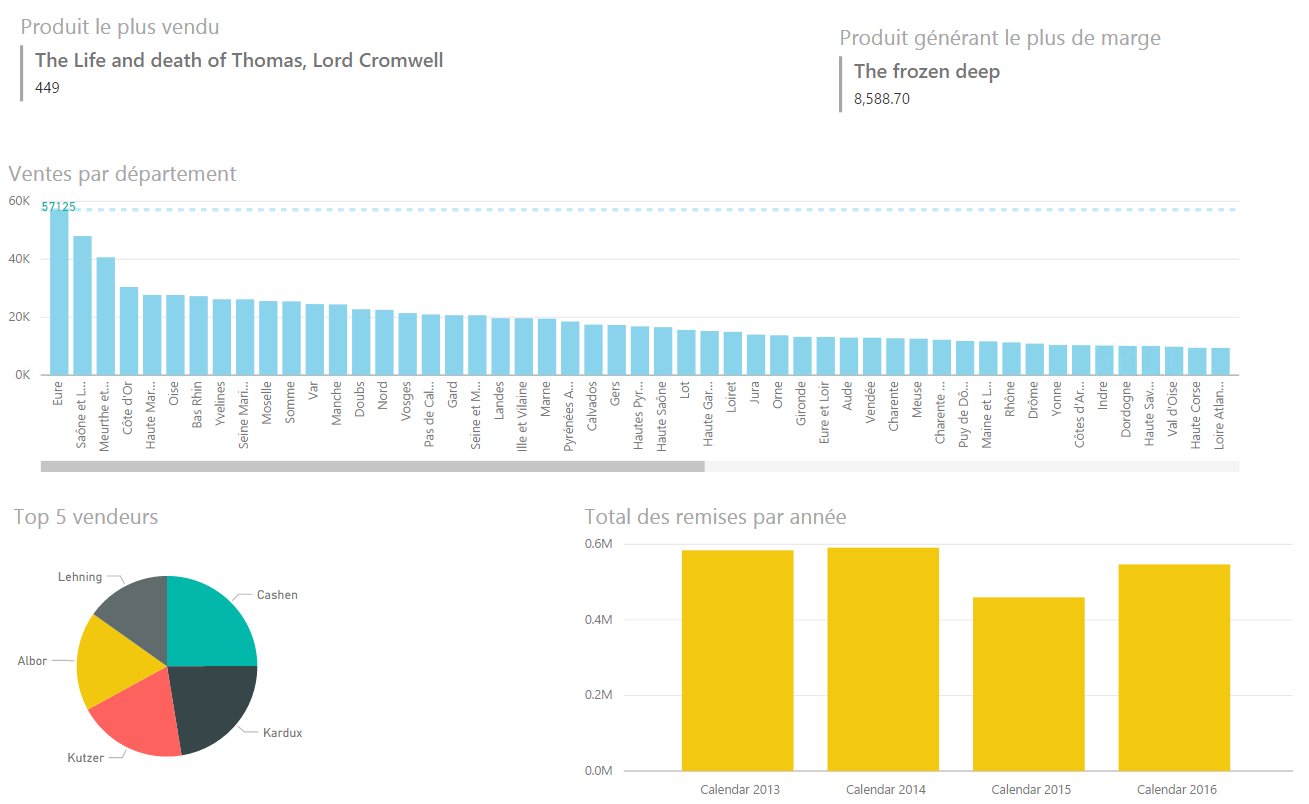
## Somme des ventes par département (région)

Mediazon distribuant dans un grand nombre de régions (départements français), nous avons jugé utile de connaître la répartition du chiffre d’affaire selon ces différentes régions.

Cela permettrait également la mise en place d’une stratégie marketing, par exemple, afin de cibler les régions où nos produits sont les plus distribués.



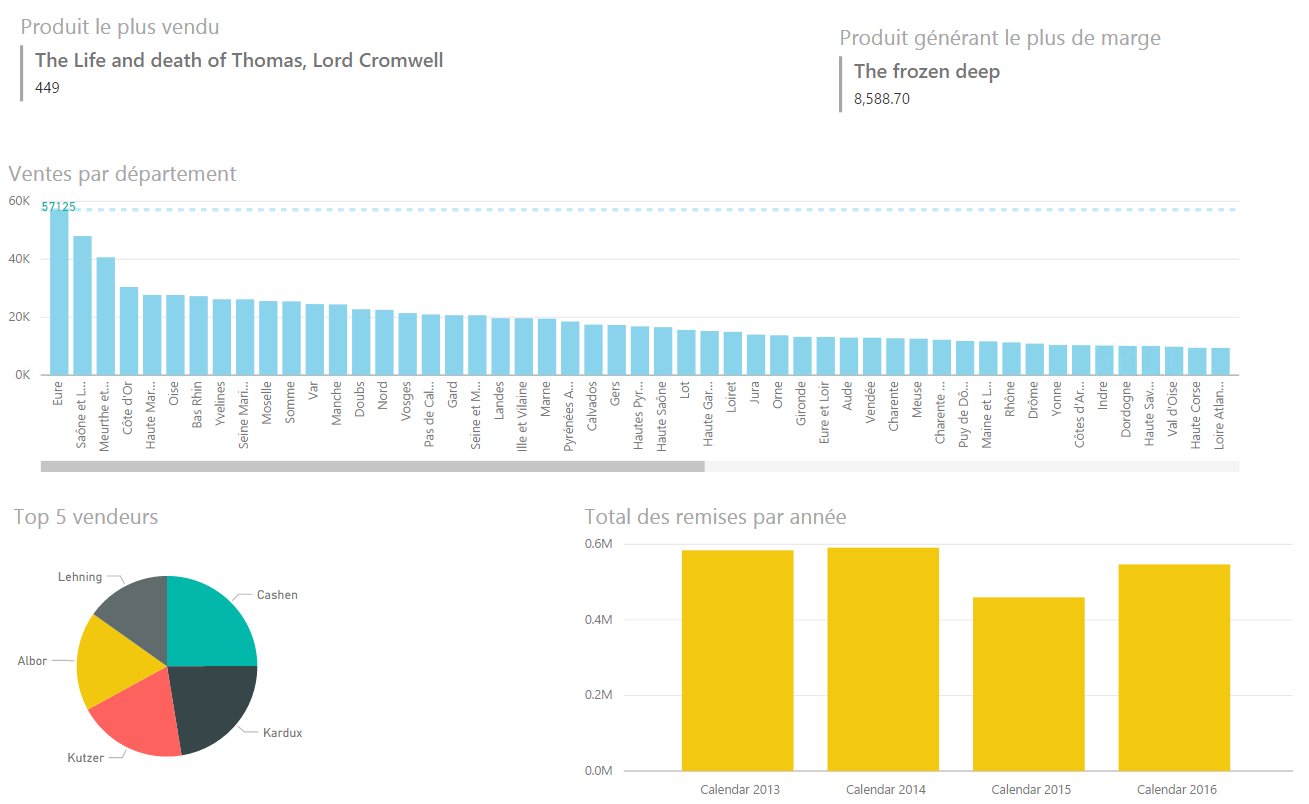
## Top 5 des vendeurs

Cet indicateur est plus « interne » que les autres : il permet de faire ressortir les cinq vendeurs ayant réalisé le plus de ventes.

Grâce à lui, l’entreprise pourrait mettre en place un système de récompenses pour les meilleurs vendeurs, ce qui permettrait d’augmenter leur motivation et leur implication dans leur travail. Il en découlerait aussi certainement une augmentation des ventes pour l’entreprise.

## Bénéfice réel par année

Non seulement nous avons créé un indicateur permettant de calculer le chiffre d’affaires par année, mais nous en avons également créé un autre qui permet de connaître la somme totale des remises par année. Cela représente donc notre bénéfice réel annuel sur l’ensemble des produits vendus.



# Conclusion

Nous avons mis en place une structure permettant de stocker les données générées par les commandes au sein de l’entreprise Mediazon. Le processus ETL a bel et bien été mis en place et été automatisé : nos processus extraient les données de leur source, les transforment et les chargent dans notre infrastructure finale.

Le système est donc fonctionnel pour autant que les données sources soient toujours au même format (mêmes types de fichiers, délimités/non délimités). Si par la suite ce format changeait, il serait nécessaire d’adapter nos processus en conséquence. Néanmoins, cela nécessiterait peu de changements, beaucoup moins que dans un processus réalisé manuellement et à chaque importation.

Les artefacts générés, comme le cube par exemple, s’adapter facilement aux besoins des utilisateurs. Si celui-ci souhaiterait générer de nouveaux indicateurs, il suffirait d’ajouter certains éléments à notre cube afin de les créer. La solution a donc été construite en pensant à l’utilisateur final étant donné que c’est lui qui utilisera le système mis en place.

Les besoins définis au début du projet ont donc été satisfaits dans leur intégralité : des contraintes ont été fixées pour les données finales qui sont désormais d’une qualité correcte malgré un certain nombre d’erreurs dans les données sources. L’importation et la transformation de celles-ci s’effectue automatiquement afin d’éviter à l’utilisateur de le faire, ce qui demanderait un temps considérable et augmenterait le nombre d’erreurs involontaires.

La gestion des erreurs a également été mise en place, accompagnée d’un contrôle et d’une reprise automatique des flux incorrects. Le système ne s’arrêtera donc pas subitement, avec un message d’erreur incompréhensible pour un non informaticien. De plus, l’ensemble des données se trouvent désormais à un seul et même endroit : le Data Warehouse.

Les indicateurs définis ont été mis en place, permettant une aide à la décision à la direction de l’entreprise ou simplement des statistiques à présenter. Grâce aux hiérarchies, les informations des commandes sont ajustables à plusieurs niveaux de granularité (année – mois – jour / catégorie – sous-catégorie – article / etc…).

Nous sommes cependant d’avis que la solution mise en place devra être améliorée et enrichie au fil du temps. Il serait utile d’améliorer constamment les processus créés en fonction des besoins, de mettre en place une intégration continue pour le système. Il est nécessaire que le projet soit continuellement mis à jour car il a beaucoup de potentiel.

# Bibliographie

GAUCHET Thomas, *SQL Server 2014 : Implémentation d'une solution de Business Intelligence*, Editions ENI, 2014

FANTINI Sébastien, *Business Intelligence avec SQL Server 2008 R2 : Maîtrisez les concepts et réalisez un système décisionnel*, Editions ENI, 2010

GASPOZ Cédric & DAVID Roxana, *Support de cours d'entrepôt de données*