

Projet BigData

**Livrable 1**

Amélie WILLEMS

Valentin DREMAUX

Pierre-Louis PERY

Amaury ROY

16/05/2021

Contexte

Le potentiel énorme associé aux données médicales a conduit le secteur de la santé à une transformation importante et rapide. Pour progresser dans la bonne voie, les praticiens (médecins, personnel infirmier) et les administrateurs d'établissements doivent pouvoir accéder directement aux informations exploitables dans les données médicales, afin d'améliorer leurs performances et la qualité des soins de manière mesurable.

le groupe CHU (Cloud Healthcare Unit), faisant partie du secteur hospitalier, as pris conscience de la nécessité d’une transformation digitale, notamment au niveau des données. Notre service est sollicité pour mettre en place leur propre entrepôt de données qui permettra au groupe d'exploiter la quantité considérable de données générées par les systèmes de gestion de soins.

Ce rapport illustre les différentes manières dont nous travaillons avec les données fournies, de façon à arriver à nos fins.

Nous verrons en détail l'architecture utilisée, et les différents programmes et phases par lesquelles nos données sont passées.

Nous étudierons en détail les données récoltées à la suite de ce livrable dans le prochain.

Table des matières

[Contexte 2](#_Toc103588159)

[Modélisation des différents axes d'analyse ainsi que les mesures 4](#_Toc103588160)

[Description de l'architecture de l’entrepôt de données 8](#_Toc103588161)

[Développement des jobs d'alimentation du schéma décisionnel 10](#_Toc103588162)

[Localisations 10](#_Toc103588163)

[Professionnel de Santé 12](#_Toc103588164)

[Diagnostiques 14](#_Toc103588165)

[Patients 15](#_Toc103588166)

[Conclusion 19](#_Toc103588167)

Modélisation des différents axes d'analyse ainsi que les mesures

Les besoins d’analyse mis en évidence par le groupe CHU sont les suivants :

* **Taux de consultation des patients dans un établissement X sur une période Y**

On doit donc pouvoir récupérer un taux de consultations décrit par rapport à un établissement et une période (ensemble de dates).

* **Taux de consultation des patients par rapport à un diagnostic X sur une période Y**

On doit donc pouvoir récupérer un taux de consultations décrit par rapport à un diagnostic et une période (ensemble de dates).

* **Taux global d'hospitalisation des patients dans une période donnée Y**

On doit donc pouvoir récupérer un taux d’hospitalisations décrit par rapport à une période (ensemble de dates).

* **Taux d'hospitalisation des patients par rapport à des diagnostics sur une période donnée**

On doit donc pouvoir récupérer un taux d’hospitalisations décrit par rapport à un diagnostic et une période (ensemble de dates).

* **Taux d'hospitalisation/consultation par sexe, par âge**

On doit donc pouvoir récupérer un taux d’hospitalisation et de consultation décrit par rapport à le sexe ou l’âge des patients.

* **Taux de consultation par professionnel**

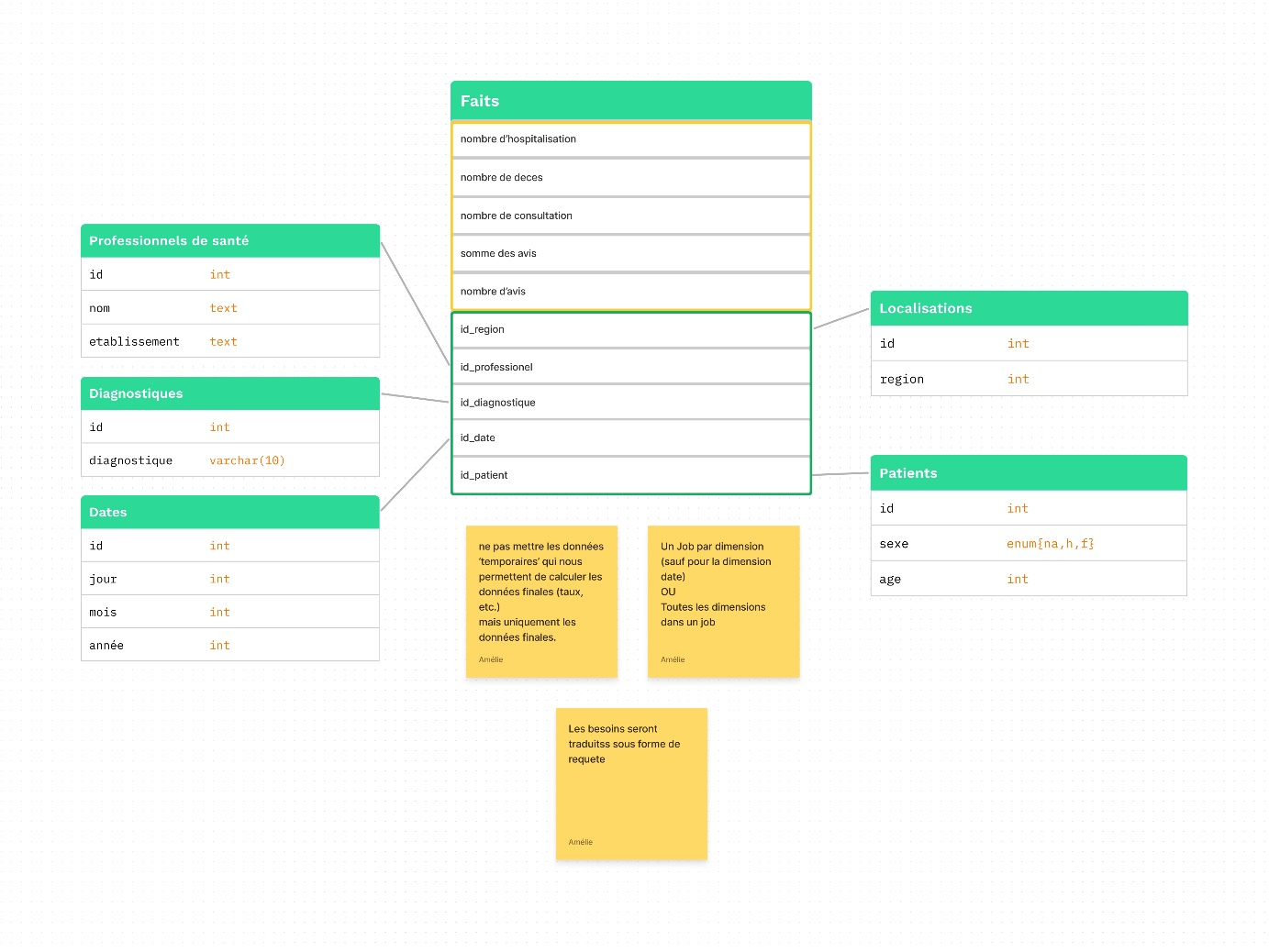
On doit donc pouvoir récupérer un taux de consultation décrit par rapport aux professionnels de santé.

* **Nombre de décès par localisation (région) et sur l'année 2019**

On doit donc pouvoir récupérer un nombre de décès décrit par rapport à une localisation et sur la période de l’année 2019.

* **Taux global de satisfaction par région sur l'année 2020**

On doit donc pouvoir récupérer un taux de satisfaction décrit par rapport à une localisation et sur la période de l’année 2020.



Ce schéma représente un modèle dimensionnel en étoile.

Le modèle en étoile a un avantage principal : Il minimise les jointures, ce qui est extrêmement important en Big Data. De plus, vu que l'on travaille avec Hive, et que Hive nous restreint sur le nombre de jointures.

Il a aussi des inconvénients, tels que la de-normalisation partielle de certaines données, mais dans notre cas, la vitesse d'exécution est plus importante que l'espace pris.

Il existe d'autre modèles, tels qu’en flocon, ou même en constellation, que l'on a jugé moins adapte à nos besoins et qui ne seront donc pas décris ici.

La table centrale est la table de fait. Elle pourrait être découpée en 2 parties :

* Vert : Liens avec les dimensions autour
* Jaune : Données calculées. Dans notre cas spécifique, elles représentent les données par jour, et par région.

Elle représente les données qui nous intéressent réellement, et sur lesquelles nous basons nos analyses.

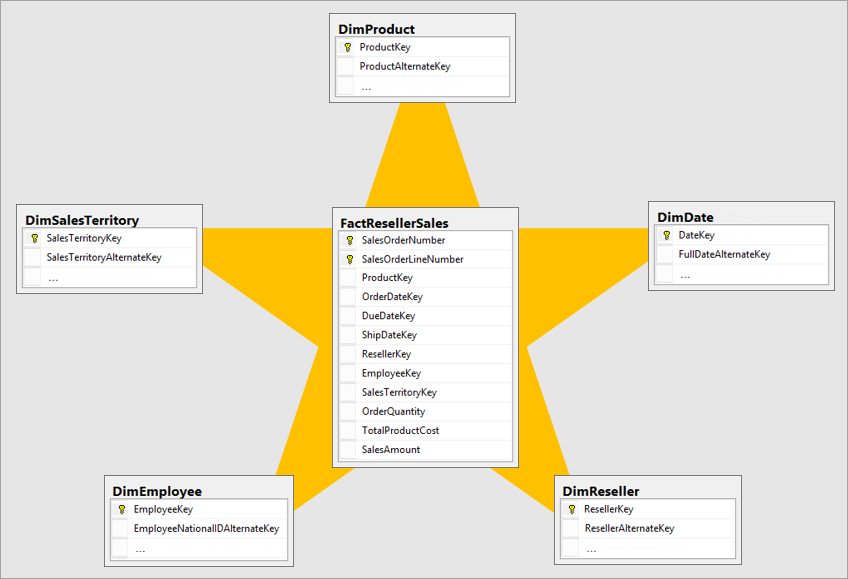
Nous avons besoins d'autre tables autour de la table de Fait, pour étendre nos relations et obtenir un contexte plus pousse pour nos analyses, tout en gardant une certaine forme de normalisation.

Chaque table autour de la table de fait est donc appelée une table de dimension, car elle donne des dimensions à nos données, elle les contextualise et les rends exploitable.

Pour donner un exemple, on va pouvoir visualiser notre nombre de décès décris par la dimension localisation où est stockée la région. Ceci répondra à notre besoin sur le « Nombre de décès par localisation (région) et sur l'année 2019 ».

Le nom de model en étoile est assez simple à comprendre avec ces informations : la table de fait représente le corps de l’étoile, et chaque dimension une de ces branches.

Ce schéma de la documentation Microsoft power BI la représente de façon très explicite :



Pour en apprendre plus sur les schémas en étoile, vous trouverez [ici](https://docs.microsoft.com/fr-fr/power-bi/guidance/star-schema) des explications plus poussées sur les nombreux avantages qu'elle nous fournit.

Dans notre cas, nous avons fait le choix d'avoir 5 dimensions:

* Professionnels de santé:
  + Taux de consultation des patients dans un établissement X
  + Taux de consultation par professionnel
* Diagnostiques
  + Taux de consultation des patients par rapport à un diagnostic X sur une période de temps Y
  + Taux d'hospitalisation des patients par rapport à des diagnostics sur une période donnée
* Dates
  + Taux de consultation des patients dans un établissement X sur une période de temps Y
  + Taux de consultation des patients par rapport à un diagnostic X sur une période de temps Y
  + Taux global d'hospitalisation des patients dans une période donnée Y
  + Nombre de décès par localisation (région) et sur l'année 2019
  + Taux global de satisfaction par région sur l'année 2020
* Patients
  + Taux d'hospitalisation/consultation par sexe, par âge
* Localisation
  + Nombre de décès par localisation (région) et sur l'année 2019

Ces dimensions sont intimement liées aux besoins des décideurs, car nous ne souhaitons ne stocker que les données nécessaires, mais il faut évidemment en stocker assez.

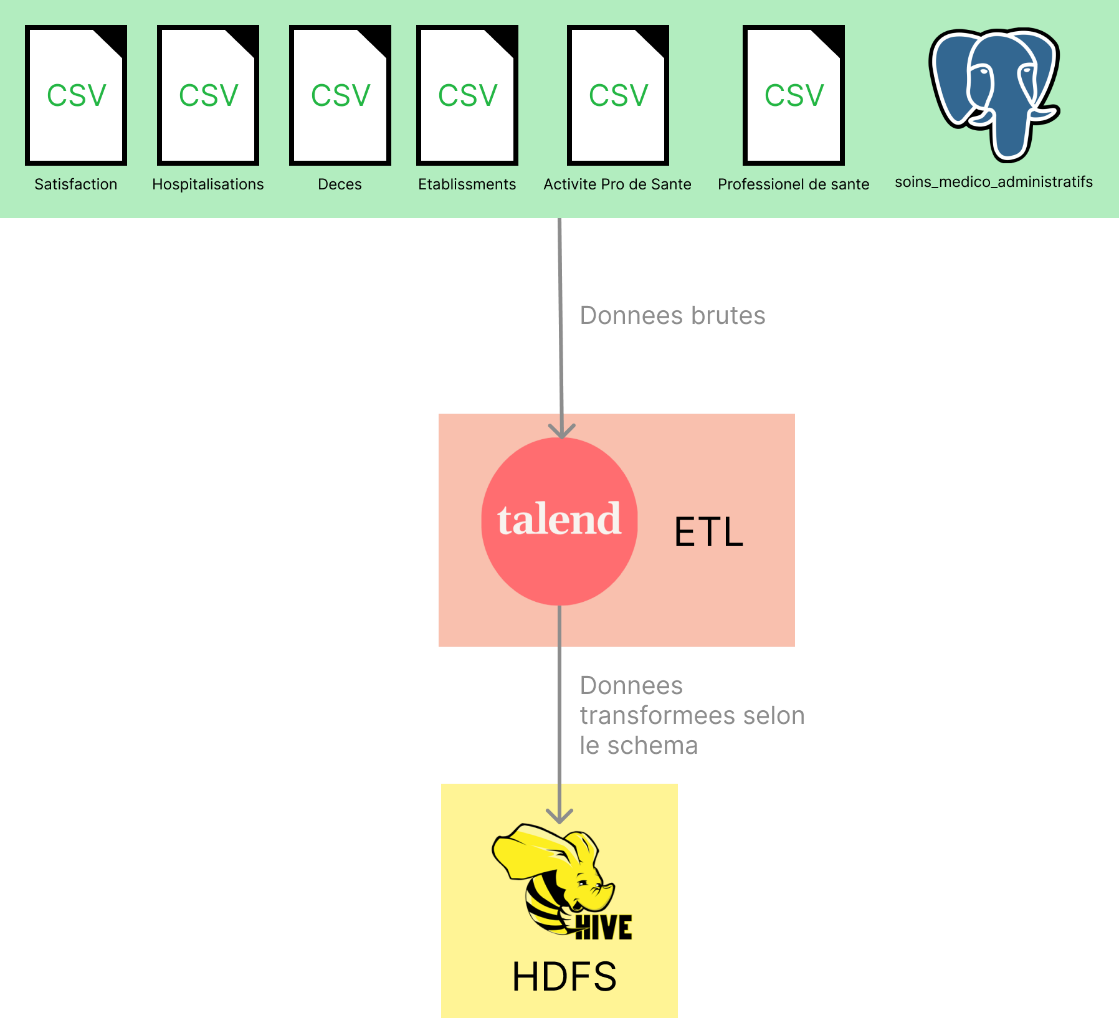
Une fois notre modèle choisis, il nous faut transformer nos données sources, très disparates, de façon a les intégrer pour pouvoir opérer dessus.

Description de l'architecture de l’entrepôt de données

L'architecture exposée ci-dessous est composée de 3 parties :

* Les sources
  + Fichiers CSV (UTF-8)
  + Fichiers Excel (xlsx, UTF-8)
  + Base de données PostgreSQL (pg://postgres:cesi@localhost:5432/soins\_medico\_administratifs)
* Les transformations
* Les sorties

Nos entrées n'ont pas de relations 1-1 avec nos sorties, et certaines sorties sont composées d'agrégats d'autres sorties, telle que la table de fait.

Nous avons donc des étapes intermédiaires. Il nous faut par exemple stocker les transformations des dimensions pour remplir la table de fait. De plus, les dimensions elles-mêmes peuvent dépendre d'autre dimensions, ou tout simplement d'autre agrégats.

Nous utilisons plusieurs étapes intermédiaires de façon a réduire les transformations nécessaires, et nous utilisons donc des fichiers délimités (CSV) en tant que cache de fortune, que l'on stocker dans l'HDFS.

Toutes les différentes transformations nous permettent de construire les différentes dimensions, avec lesquelles nous pouvons créer notre table de fait finale.

En fonction de nos besoins, la construction de notre table de fait va différer. Ici, nous avons fait le choix de stocker les faits en fonction de 3 dimensions clefs: **jour**, **sexe** et **région**, qui vont nous permettre d'avoir des mesures assez précise sans avoir trop d'impact sur le temps d'exécution de nos requêtes plus tard.

Ces dimensions choisies ont aussi un autre rôle : il nous est possible de partitionner les tables dans Hive avec ces dimensions, ce qui nous permet encore de réduire les temps de requêtes. L'avantage est donc double, puisque nous gagnons du temps d'exécution, et que nous n'avons que très peu de transformations a effectuer pour le partitionnement.

Développement des jobs d'alimentation du schéma décisionnel

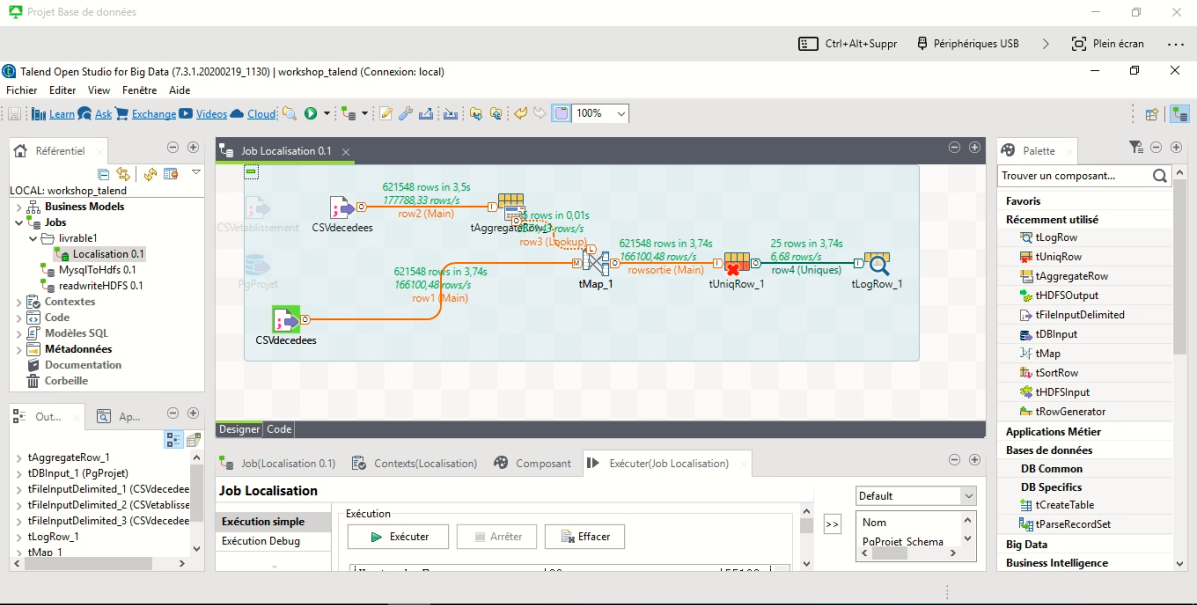
Ci-dessous, des exemples de tâches effectuées par l'ensemble de l’équipe :

### Localisations

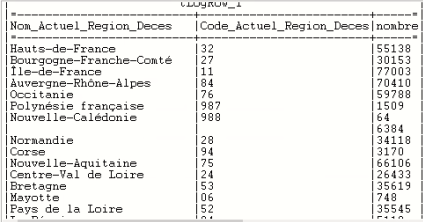
Le job localisation permet d'avoir les informations du nombre de personnes décédées ou le taux de satisfaction par régions.

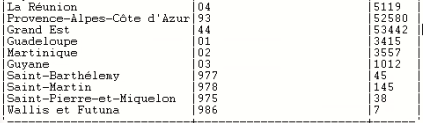
Exemple pour les décès par régions : On récupère le nombre de personnes décédés et les régions de leurs décès dans les CSV ( ici 2019 ). Puis on additionne les doublons dans le tAggregate.

Dans le tMap on fait sortir que le Nom et le Code de la région avec le nombre de personne décédés par région. On utilise Uniqrow pour obtenir une seul fois la ligne de la région dans le tableau.



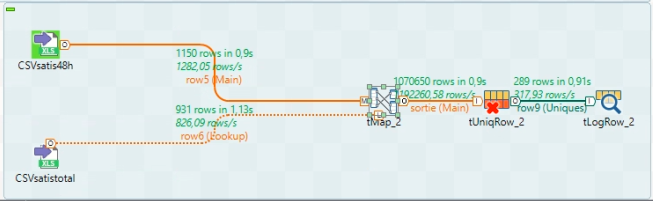
on obtient donc :

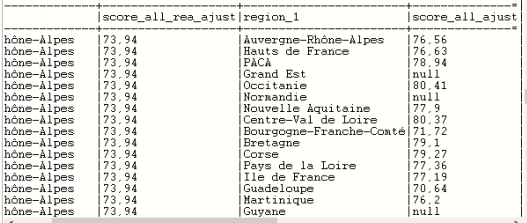




Pour le taux de satisfaction par région il faut faire la moyenne par région des taux globaux des hospitalisation moins de 48heures ou plus 48 heures.

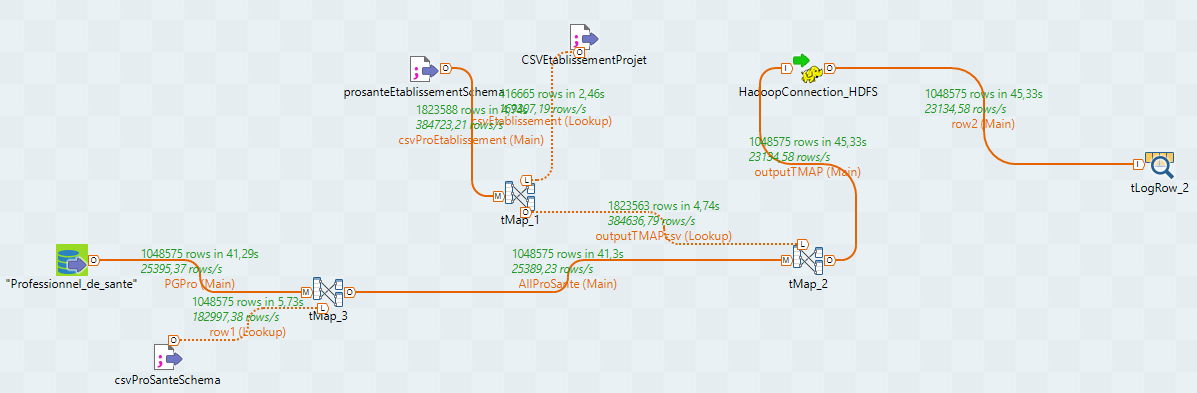
On importe dans le tMap les résultats par régions que l'on fait ressortir puis on évite d'avoir des doublons de régions dans le Uniqrow pour obtenir qu'une seule fois chaque région avec la moyenne correspondante a cette dernière.



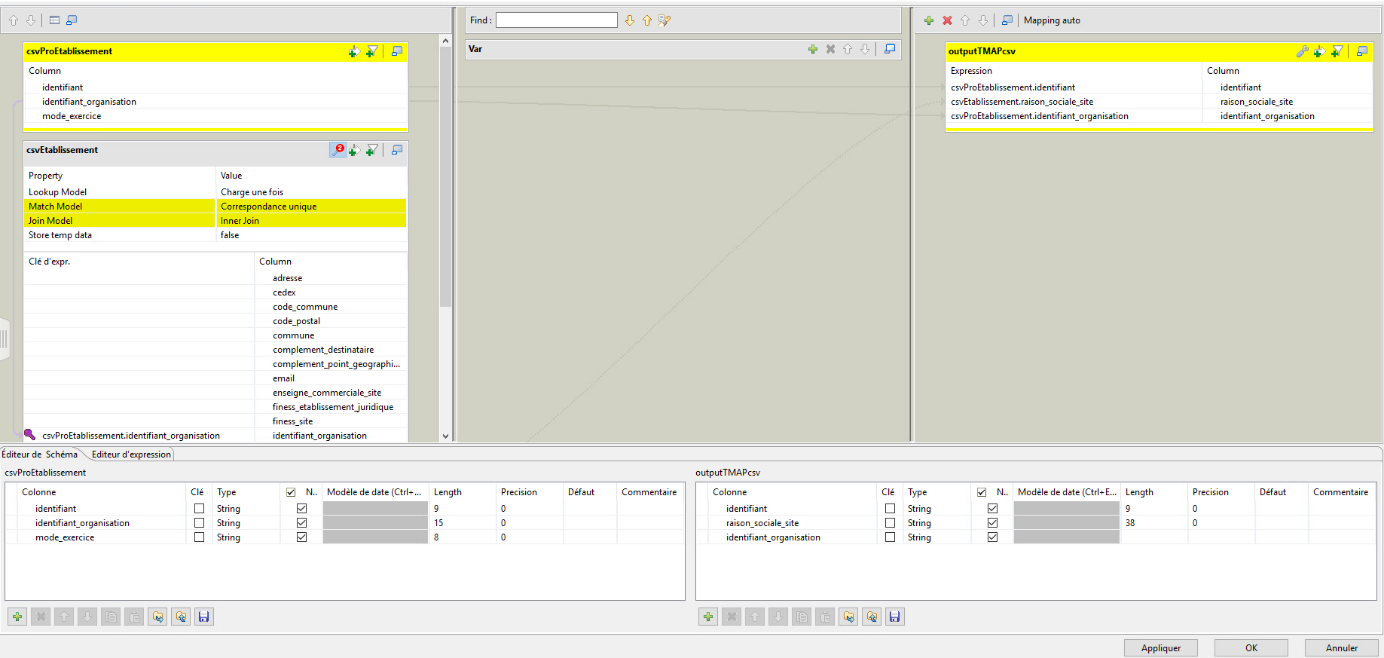


### Professionnel de Santé

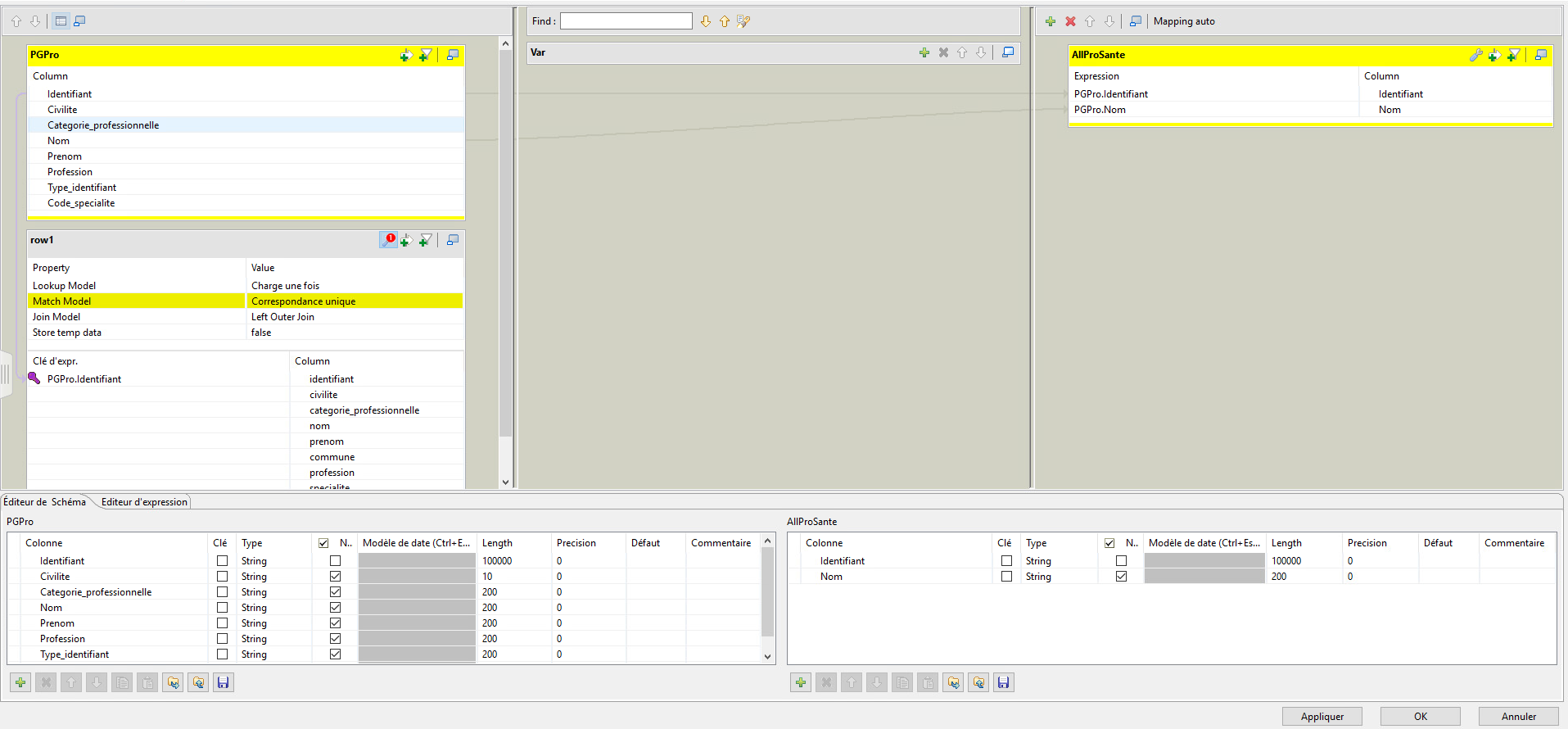
Le job Professionnels de santé permet de remplir notre dimension Professionnels de santé qui contient un id généré, le nom du praticien, et l'établissement de santé auquel il est relié.



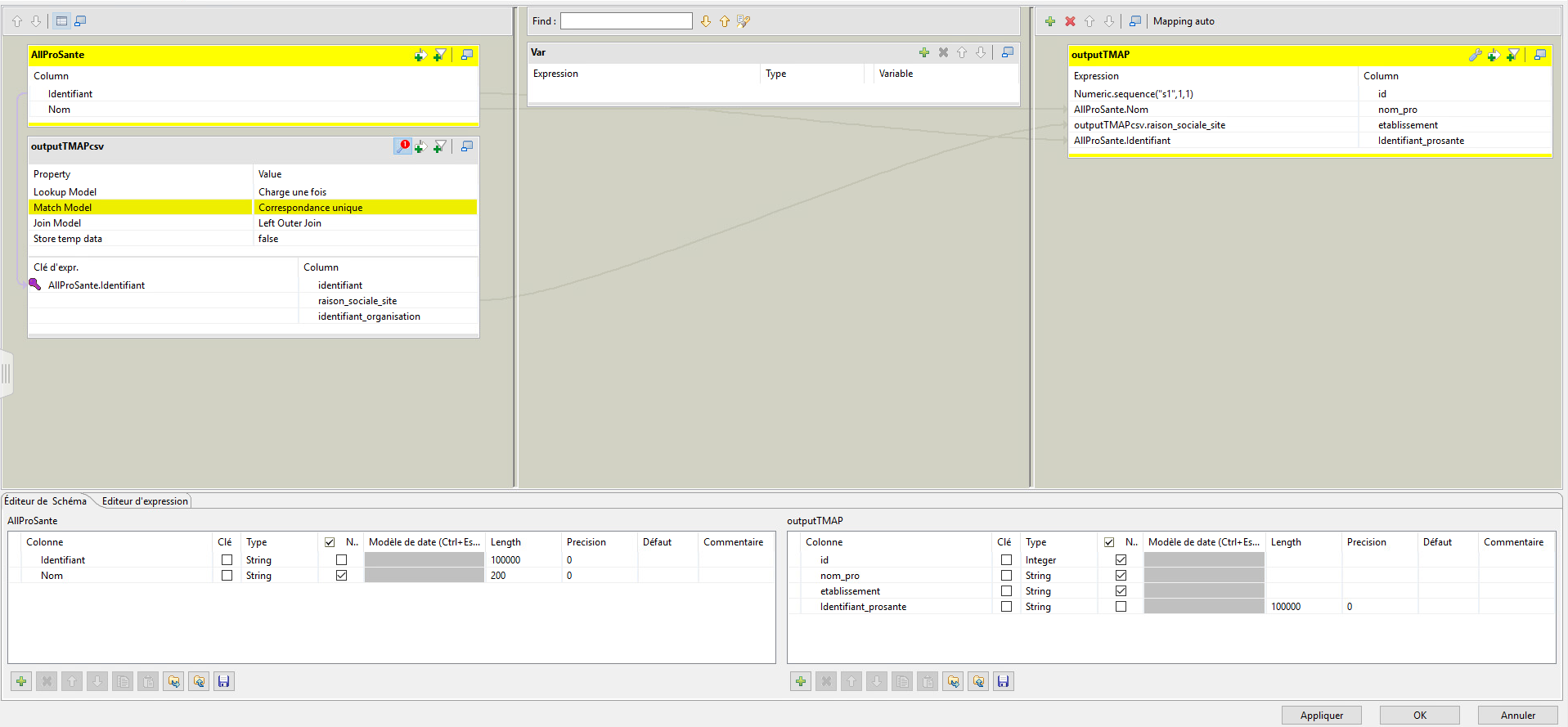
On récupère les informations de différentes sources : les fichiers CSV et la base de données PostgreSQL. Tout d'abord (en haut, au milieu du schéma) on récupère les informations des CSV "etablissement\_sante" et "activite\_professionels\_sante" que l'on joint avec un tmap pour récupérer les identifiants des établissements qui sont reliés à des professionnels de santé, les identifiants des professionnels de santé et la raison sociale de l'établissement (équivalente à son nom)



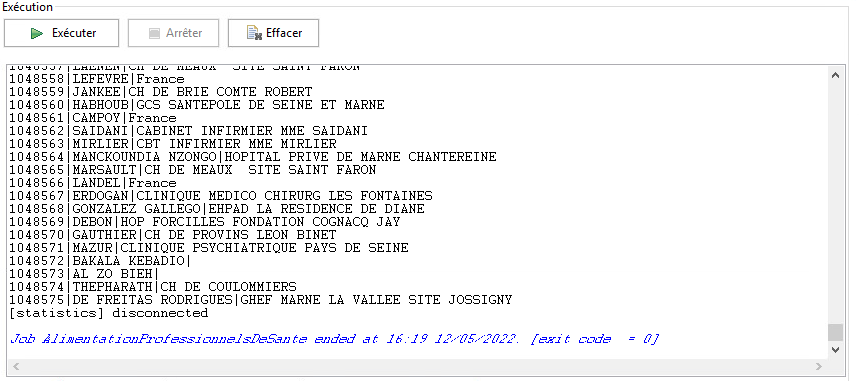
Ensuite on se connecte à la table des professionnels de santé de PostgreSQL et on récupère les informations du CSV des professionnels de santé, et on joint les deux pour récupérer tous les professionnels de santé. On part du principe que certaines informations peuvent différer dans les fichiers CSV et PostgreSQL et on joint donc les deux pour bien récupérer toutes les informations. A l'aide du Tmap on trie les informations nécessaires et on ne garde que l'identifiant et le nom du professionnel de santé.



Ensuite on joint nos deux output tmap pour pouvoir générer notre dimension finale avec un identifiant autogénéré, le nom du praticien et l'établissement dans lequel il exerce (si le praticien est en libéral, la colonne établissement reste vide).



On enregistre ensuite le tout dans HDFS, et on affiche notre résultat dans un LogRow.



### Diagnostiques

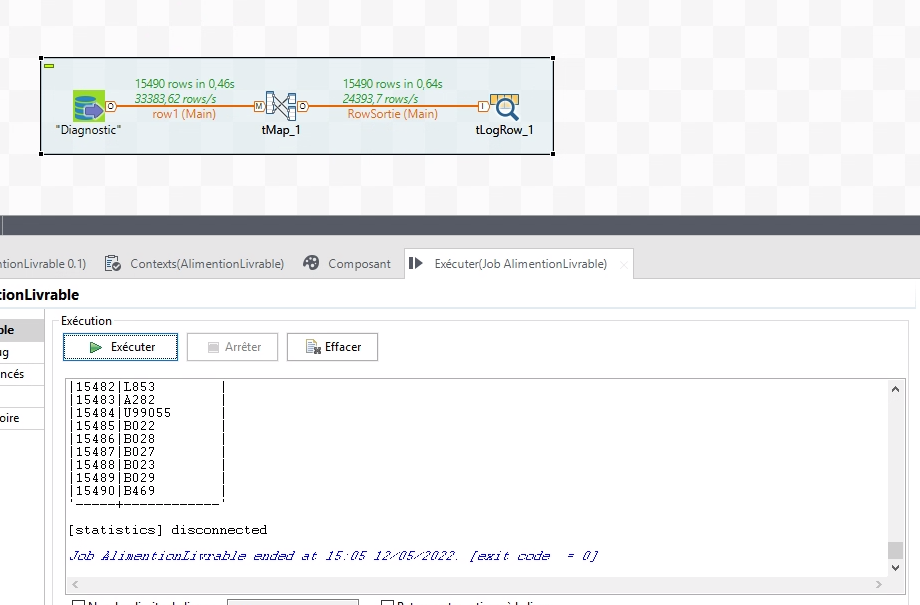
Le job dans Talend, ne pas oublier d'enlever tout le surplus dans la requête.

"SELECT

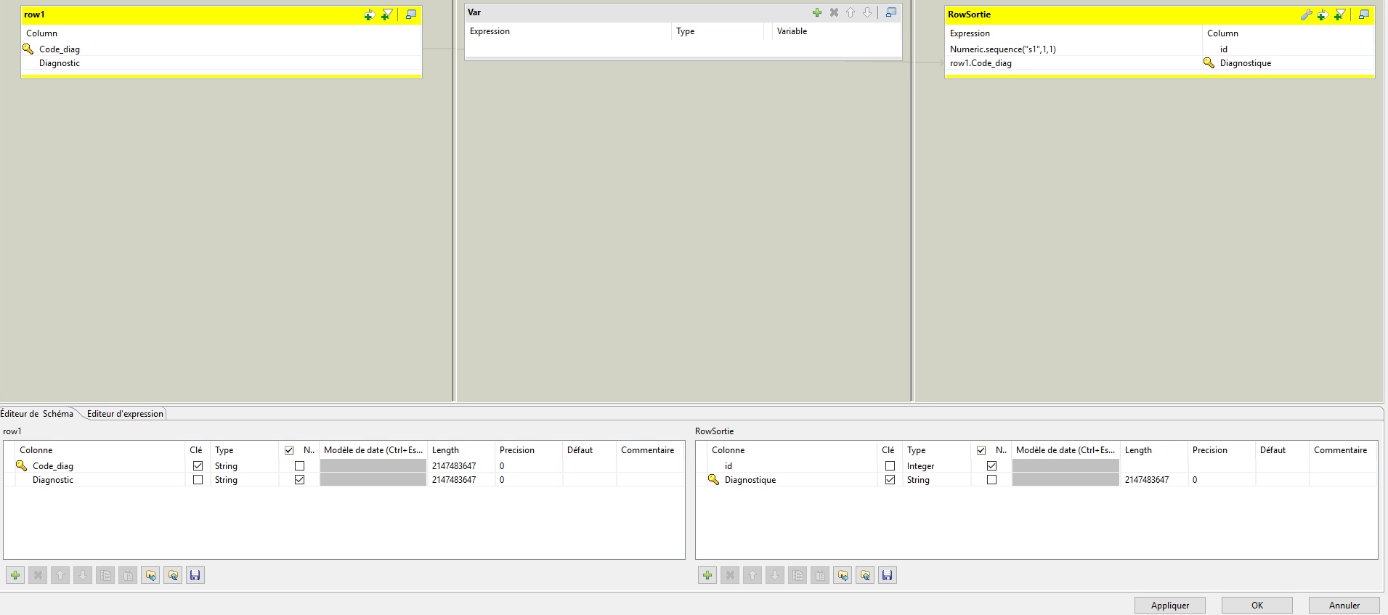
\"Code\_diag\",

\"Diagnostic\"

FROM \"Diagnostic\""



tMap du job Diagnostic:



### Patients

SQL:

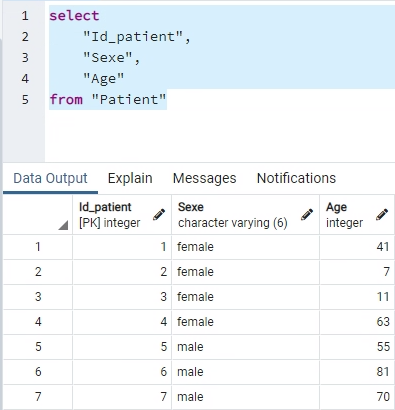
SELECT

"Id\_patient",

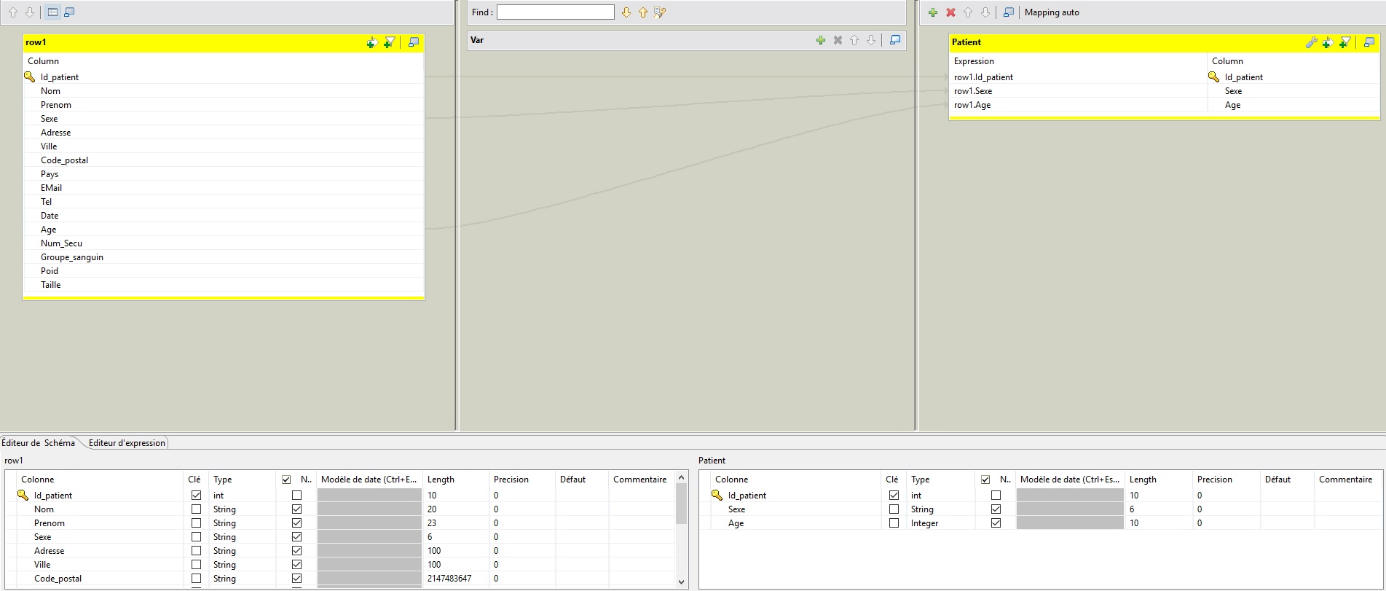
"Sexe",

"Age"

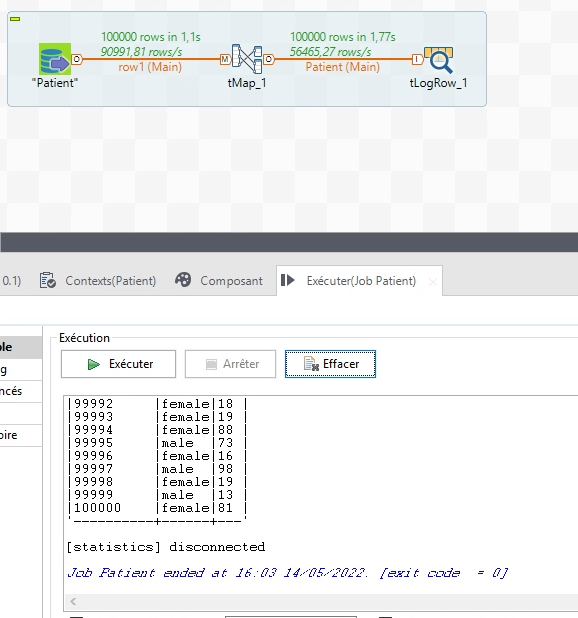
FROM "Patient"



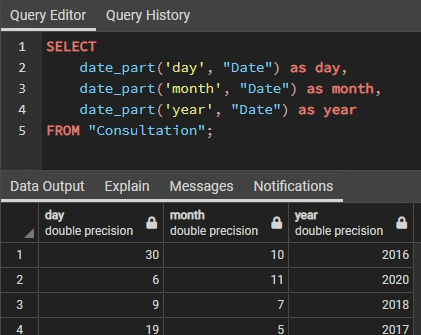
Le mapping:

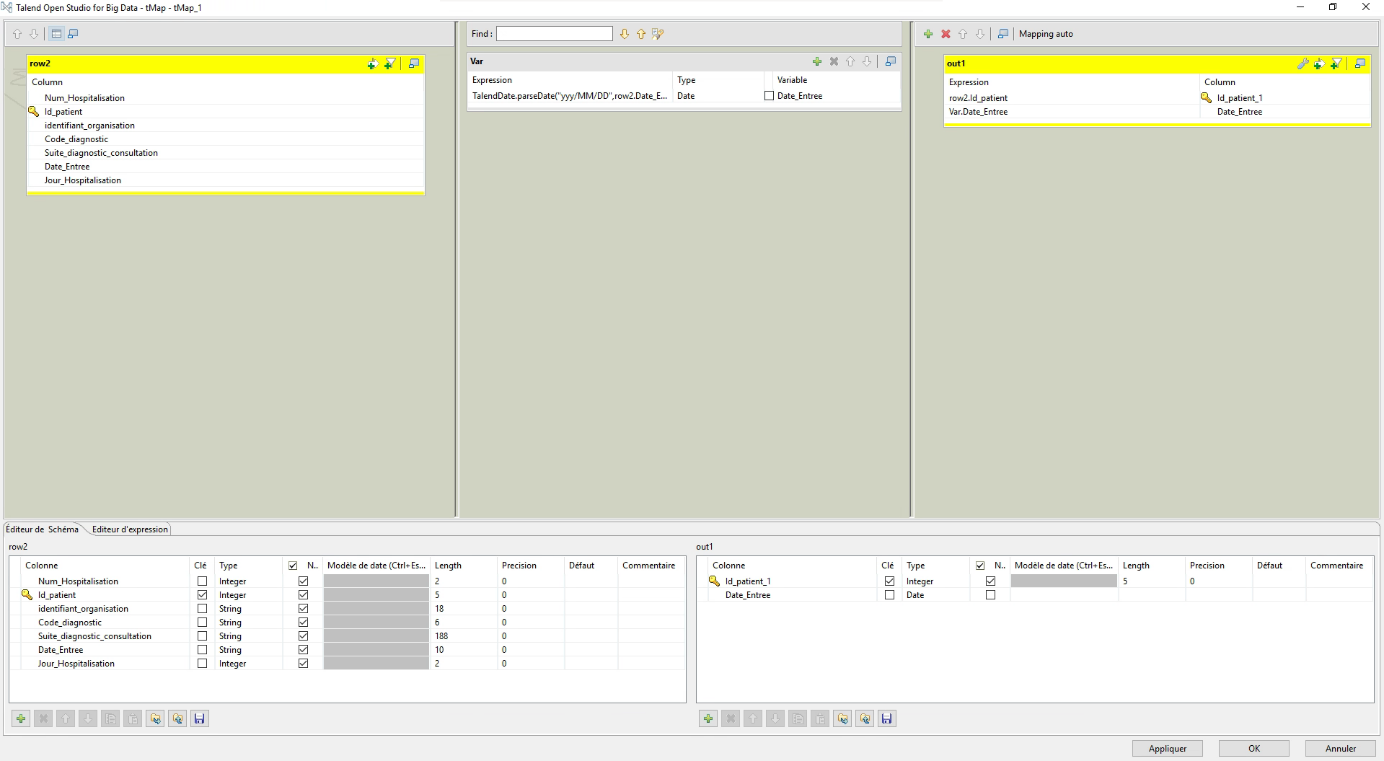


Le résultat :



Evidemment nous avons fait d'autre tâches intermédiaires pour remplir la table de fait, et d'autres dimensions "secondaires" telles que les dates,





Pour lesquelles nous avons besoins de plusieurs sources de données différentes à agréger.

Conclusion

Nous avons commencé par analyser nos différents besoins pour définir notre modèle décisionnel en étoile. Ce modèle nous permet de répondre à chacun des besoins en calculant une valeur dans la table de fait, valeur qui peut être décrite par une ou plusieurs dimensions en fonction de nos besoins. Ensuite nous avons décrit l’architecture de notre entrepôt de données, où seront stockées nos données. Pour terminer, nous avons développé nos jobs en utilisant talend, ces jobs remplissent nos dimensions. On utilise dans ces jobs les données mises à notre disposition dans les différents fichiers CSV, XSLX, et la base de données postgres, et nous faisons des traitements sur ces données pour qu’elles soient triées et utilisables dans notre modèle décisionnel.