

PROJET BIG DATA

LIVRABLE 1

FODIL Nel | MARCELLI Enzo | GOUADFEL Rayan
12 avril 2024

Table des matières

I.	Intro	3
II.	Contexte du projet.....	3
III.	Utilisation de la Big Data dans le projet.....	4
IV.	Planification Projet.....	5
V.	Architecture décisionnelle	6
VI.	Dimensions et Tables.....	10
VII.	Modèle de Données	13
1.	Vue Modèle Conceptuel des Données (MCD).....	13
2.	Vue Modèle Logique de Données (MLD).....	18
VIII.	Job d'alimentation.....	19
IX.	Automatisation des jobs.....	23
X.	Conclusion	26
	Webographie	27

Table des figures

Figure 1 : Big Data 5V https://medium.com/@get_excelsior/big-data-explained-the-5v-of-data-ae80cbe8ded1	4
Figure 2 : GANTT	5
Figure 3 : Architecture décisionnelle	8
Figure 4 : MCD Etoile Consultation	14
Figure 5 : MCD Etoile Hospitalisation	15
Figure 6 : MCD ProfessionnelSanté	16
Figure 7 : MCD Décès	16
Figure 8 : MCD Global	17
Figure 9 : Modèle Logique de Données	18
Figure 10 : Exemple Job d'Alimentation	19
Figure 11 : tMap EtablissementSante	20
Figure 12 : Job Patient	20
Figure 13 : Ensemble des jobs d'alimentation	21
Figure 14 : Job Master	22
Figure 15 : Fichiers d'Alimentation du Datalake.....	23
Figure 16 : Planificateur de tâches	24
Figure 17 : Script job master .bat	25

I. Intro

Avec l'augmentation des données médicales, le groupe CHU (Cloud Healthcare Unit) cherche à transformer numériquement ses opérations. Leur objectif est d'améliorer la gestion et la qualité des soins en utilisant un système avancé de gestion de données.

II. Contexte du projet

Le secteur de la santé doit adapter ses méthodes à la nouvelle ère du numérique pour rester efficace. Le groupe CHU prévoit de construire un entrepôt de données pour mieux gérer et analyser les informations provenant de diverses sources, comme les dossiers des patients et les retours sur la satisfaction des soins. En consolidant ces données, le CHU espère mieux comprendre et améliorer les pratiques de soins. Pour y parvenir, ils développent une infrastructure informatique flexible capable de traiter de grands volumes de données rapidement et de manière sécurisée.

III. Utilisation de la Big Data dans le projet

La Big Data dans le contexte du groupe CHU incarne un levier de transformation stratégique et un catalyseur d'innovation. Son utilisation ouvre de nouvelles dimensions dans la gestion de la santé, notamment :

- Amélioration de la prise en charge des patients : L'analyse de volumes importants de données permet d'identifier des schémas et des corrélations qui conduisent à une meilleure compréhension des pathologies et des traitements efficaces, offrant ainsi une prise en charge plus précise et personnalisée.
- Optimisation opérationnelle : La Big Data permet de déceler des inefficacités opérationnelles et d'optimiser la gestion des ressources des établissements de santé, ce qui mène à des économies significatives et à une allocation plus judicieuse des ressources.
- Prise de décisions basée sur des données probantes : Avec l'accès à une grande quantité de données historiques et actuelles, le groupe CHU peut développer une approche de prise de décision basée sur des preuves tangibles, réduisant ainsi les incertitudes et améliorant les stratégies de soins.
- Recherche et développement : L'accumulation et l'analyse de données variées contribuent à la recherche médicale en fournissant une base solide pour les études épidémiologiques, le développement de nouveaux traitements et la compréhension des tendances de santé publique.

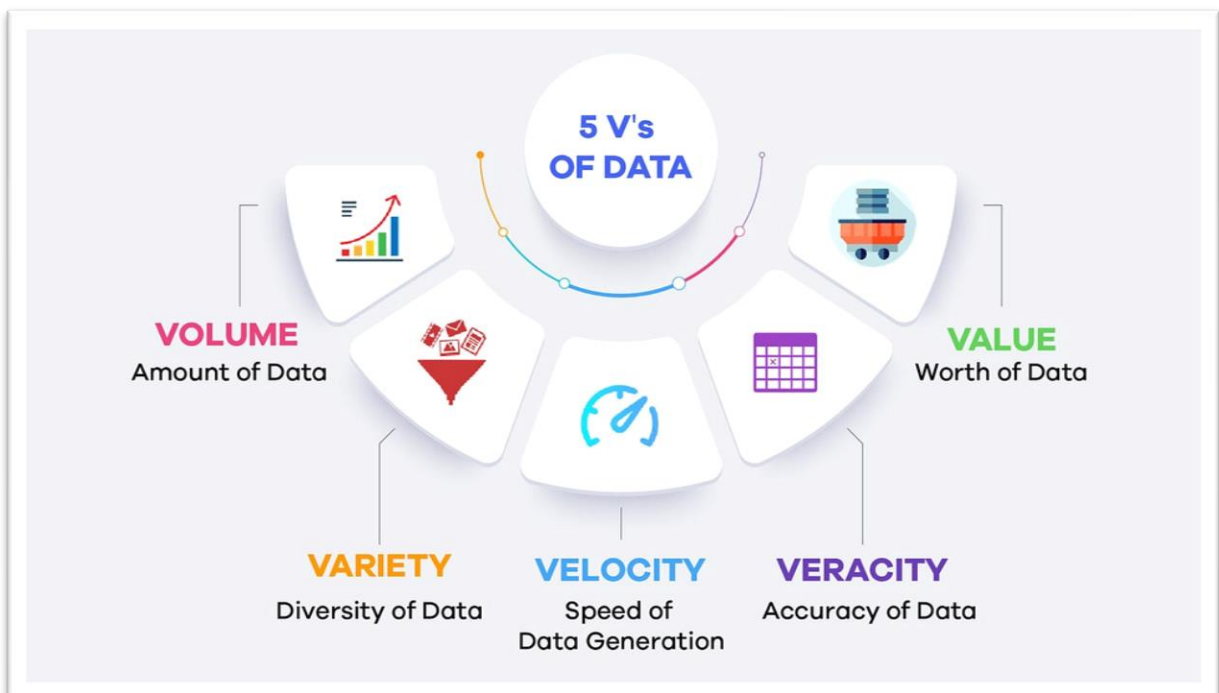


Figure 1 : Big Data 5V

https://medium.com/@get_excelsior/big-data-explained-the-5v-s-of-data-ae80cbe8ded1

IV. Planification Projet

La planification de notre projet CHU Big Data s'étend sur un mois, débutant par la mise en place du projet et l'organisation de l'environnement de travail. Nous consacrons la première semaine à la compréhension du cahier des charges et à la préparation technique. La deuxième semaine est réservée à la modélisation des données et à l'initialisation des jobs d'alimentation, suivi de l'élaboration du premier livrable. La troisième semaine se focalise sur l'optimisation du modèle physique et l'évaluation des performances. Enfin, la dernière semaine est dédiée au peaufinage du second livrable et aux préparatifs de la présentation finale.

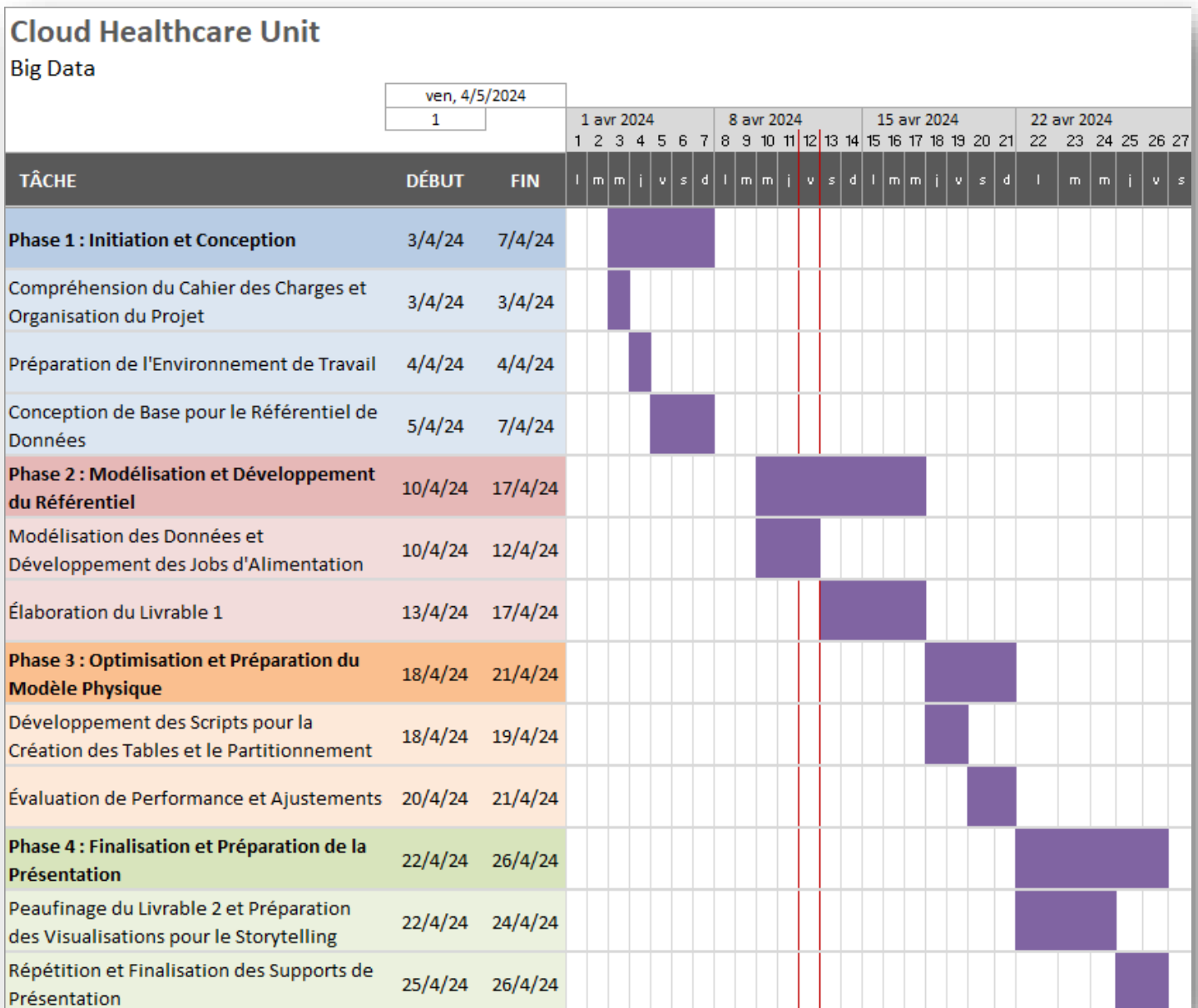


Figure 2 : GANTT

V. Architecture décisionnelle

Avant de vous présenter notre schéma d'architecture décisionnelle (SAD), explorons les outils logiciels et les concepts clés qui seront utilisés dans notre projet :

- **Hadoop** est un framework open-source conçu pour le stockage et le traitement distribué de grandes quantités de données. Il offre une infrastructure permettant de gérer efficacement les données à grande échelle, en utilisant un modèle de calcul distribué. On retrouve dans Hadoop de nombreux outils :
 - **Hadoop Distributed File System (HDFS)** : HDFS est le système de fichiers distribué de base de Hadoop. Il est conçu pour stocker de grands volumes de données sur des clusters de serveurs, en les répartissant de manière redondante pour une haute disponibilité et en les traitant de manière parallèle.
 - **MapReduce** : MapReduce est un modèle de programmation et un système de traitement par lots pour l'analyse de données distribuées. Il divise les tâches en une série de tâches Map (qui traitent les données en parallèle) et Reduce (qui agrègent les résultats), permettant un traitement efficace des données massives.
 - **Hive** : Hive est une infrastructure de traitement de données construite au-dessus de Hadoop qui permet l'interrogation et l'analyse de grands ensembles de données à l'aide d'une syntaxe similaire à SQL. Il traduit les requêtes en tâches MapReduce exécutées sur le cluster Hadoop.
- **PostgreSQL**, quant à lui, est un système de gestion de base de données relationnelle robuste et extensible, également open-source. Il est largement utilisé pour stocker et interroger des données structurées, offrant des fonctionnalités avancées telles que le support du langage SQL, la gestion des transactions et la réplication.
- **Power BI** (PBI) est une plateforme de business intelligence développée par Microsoft, permettant aux utilisateurs de visualiser et partager des insights à partir de données. Elle offre des fonctionnalités avancées de création de rapports interactifs, de tableaux de bord dynamiques et d'analyses approfondies, facilitant ainsi la prise de décisions informées au sein des organisations.
- **Talend** est une plateforme puissante dans le domaine du Big Data, offrant des outils avancés pour l'intégration, la transformation et le traitement de grandes quantités de données. Avec Talend, on peut facilement collecter des données à partir de sources variées telles que des bases de données, des applications

cloud et des flux de données en temps réel. Cette plateforme permet également de mettre en œuvre des workflows (jobs) pour orchestrer et traiter le flux de données à travers différentes étapes du processus d'analyse.

- Un **Data Lake (DLK)** est une plateforme de stockage pour des données brutes en provenance de diverses sources, sans nécessité de les structurer préalablement. Contrairement à un Data Warehouse, il ne contraint pas les données à un format spécifique. Cela permet une flexibilité et une exploration aisée des données pour les analyses.
- Un **Data Warehouse (DWH)** est une plateforme utilisée pour collecter et analyser des données en provenance de multiples sources hétérogènes. Le Data Warehouse est généralement séparé de la base de données opérationnelle d'une entreprise.
- Un **Data Mart** est une base de données spécialisée répondant aux besoins d'un département ou d'une fonction commerciale spécifique. Contrairement à un entrepôt de données qui stocke toutes les données de l'entreprise, un datamart contient des données pertinentes pour un usage spécifique. Il est souvent utilisé comme base pour des solutions de visualisation de données.
- **ELT :**
 - Extract (Extraction) : Durant cette phase, les données sont extraites de différentes sources pour les rassembler au même endroit.
 - Load (Chargement) : Une fois les données extraites, elles sont chargées directement dans un entrepôt de données ou une base de données dédiée, sans subir de transformations majeures préalables. Cette étape implique simplement le transfert des données brutes vers un espace de stockage où elles seront disponibles pour des analyses ultérieures.
 - Transform (Transformation) : Après le chargement des données dans l'entrepôt, des processus de transformation sont appliqués. Cela peut inclure des opérations telles que le nettoyage des données, la normalisation, la déduplication et d'autres transformations nécessaires pour préparer les données à l'analyse. Ces transformations sont effectuées après le chargement des données, d'où le terme "Transform" dans ELT.
- **ETL :**
 - Extract : Durant cette phase, les données sont extraites de différentes sources pour les rassembler au même endroit.
 - Transform : Une fois extraites, les données subissent des transformations pour les nettoyer, les harmoniser et les préparer à l'analyse, en appliquant des règles de validation, de conversion et de fusion.

- Load : Enfin, les données transformées sont chargées dans un entrepôt de données / base de données dédiée, où elles sont prêtes à être utilisées pour des analyses et des rapports.

Maintenant que nous avons clarifié ces concepts et outils fondamentaux, examinons plus en détail notre architecture décisionnelle :

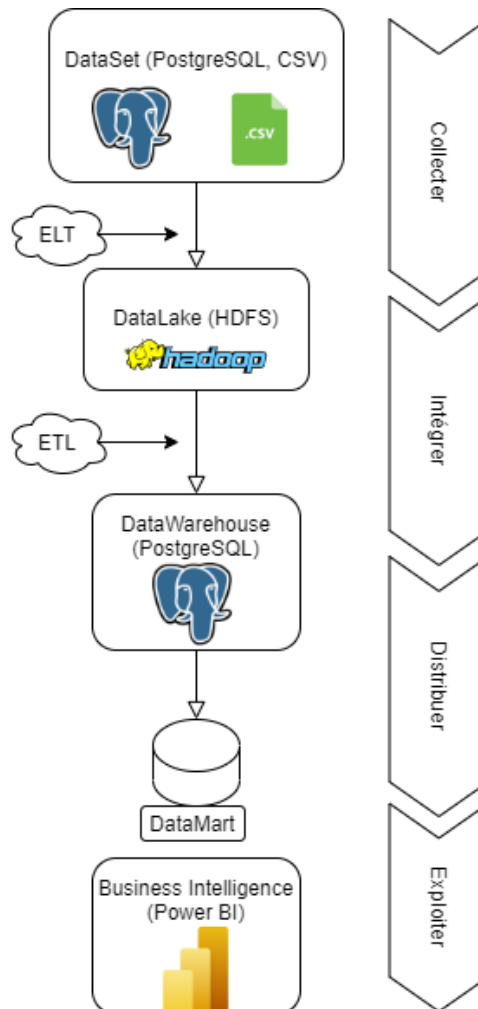


Figure 3 : Architecture décisionnelle

Notre Architecture de système d'information décisionnel comprend **7 principaux points** :

- 1- Dans un premier temps à lieu la collecte/extraction des données sources, dans notre cas nos données sources proviennent de fichier .csv ainsi que d'une base de données PostgreSQL.

- 2- Dans un second temps nous avons mis en place un processus ELT, nous avons fait ce choix pour plusieurs raisons. L'objectif d'un DLK et nous expliquerons son choix plus tard également, est de stocker des données brutes, c'est-à-dire non structuré essentiellement. Cela sous-entend qu'aucun traitement de donnée est nécessaire pour l'alimenter. L'ELT nous permet cela, en chargeant les données brutes dans le DLK sans transformation initiale. Cela simplifie le processus, réduit la complexité et accélère le chargement des données. Les transformations sont effectuées ultérieurement, selon les besoins spécifiques d'analyse ou de traitement. Cette approche permet de stocker rapidement de grandes quantités de données, puis de les transformer au fur et à mesure des besoins émergents, sans avoir à refaire le processus complet de chargement. Offrant ainsi une plus grande flexibilité pour les analyses futures. L'ELT se révèle être donc le processus adapté à l'alimentation du DLK.

- 3- Dans un troisième temps, nous avons notre DLK qui se fait alimenter de données par nos ELT. Le DLK est important car il nous permet d'éviter les désavantages de travailler directement depuis les sources de données. Permettant que nos traitements ne soient pas directement impactés par les changements ou les pannes des sources de données, offrant une plus grande fiabilité et robustesse dans notre pipeline de données. Aussi, Travailler depuis un DLK offre une flexibilité dans la façon dont nous traitons et analysons les données. Nous pouvons expérimenter avec différents types de traitements, de modèles et d'algorithmes sans risque de perturber les sources de données ou d'impacter les performances des systèmes opérationnels. Également, en recopiant les données dans le DLK, on conserve une historisation des données. Ce qui permet de réaliser des analyses rétrospectives et de remonter le temps pour comprendre l'évolution des données et des performances. Le DLK sert de base commune, facilitant l'accès aux données pour l'ensemble de l'organisation. Et finalement, on retrouve des technologies de traitement distribué permettant d'accélérer les traitements et améliorer les performances

- 4- Dans un quatrième temps, se trouve en sortie du DLK notre processus ETL. A la différence des ELT, l'objectif va être de transformer les données en un format adapté à nos besoins pour le chargement dans le DWH. Cela peut inclure la normalisation, l'enrichissement et la consolidation des données pour une analyse plus efficace. Cette partie est propre aux besoins de chacun, donc un processus ETL ne ressemblera pas forcément à un autre.

- 5- Dans un cinquième temps, après avoir réalisé nos traitements sur les données par le biais des ETL nous chargeons nos données transformées dans le DWH. Ce dernier sert de base de données pour stocker les données « finales », qui ont subi les transformations nécessaires pour enfin être analysées. Il permet une vue unifiée et cohérente des données pour les analyses et les rapports, ce qui facilite la prise de décision. C'est depuis le DWH qu'on va alimenter les différents Data Mart, cependant tout dépend de l'architecture souhaitée. Il est aussi possible de réaliser des analyses ou utiliser des outils de visualisation directement depuis le DWH.

- 6- Dans un sixième temps, on retrouve des Data Mart. Ce dernier, est une BDD dont le contenu est en rapport avec une activité ou un secteur de l'entreprise. Cela permet de mettre à disposition les données de manière privative en fonction de l'activité. Par exemple les données RH ne sont pas censées être visible par une autre activité de l'entreprise, et inversement. En segmentant les données dans plusieurs Data Mart (donc BDD), les performances des requêtes analytiques sont généralement améliorées. Contrairement au DWH, il répondra seulement aux besoins d'un département donné ou d'une fonction métier spécifique. C'est depuis les Data Mart qu'on peut aussi mettre en place des solutions de visualisation de données, qui est notre dernier point.

- 7- Finalement, on retrouve au bout de la chaîne la visualisation de données. Elle permet de donner vie aux données, afin d'aider à la prise de décision. Nous allons utiliser Power BI mais il existe d'autres outils comme Click View. Les outils de visualisation de données offrent une interface utilisateur interactive qui permet aux utilisateurs de créer des tableaux de bord personnalisés et de naviguer facilement à travers les données. Cela permet une exploration approfondie des données et une découverte de nouveaux insights. Les graphiques, les tableaux croisés dynamiques, les cartes et autres éléments visuels permettent de représenter les données de manière efficace et intuitive. Cela facilite la communication des résultats d'analyse et la compréhension des tendances et des modèles

VI. Dimensions et Tables

Dans un modèle de données dimensionnelles, les tables de faits et les tables de dimensions sont deux types de tables distincts qui fonctionnent ensemble pour organiser les données d'un entrepôt de données de manière optimisée pour l'analyse.

Une table de faits est une table principale dans un modèle dimensionnel, elle contient des colonnes clés de dimension qui se rapportent aux tables de dimension et des colonnes concernant des attributs propre à la table de faits.

Une table de dimensions contient les dimensions d'un fait, ils sont donc joints à la table de faits via une clé étrangère. Les tables de dimensions représentent des extensions d'une table de faits. Dans l'exemple du projet, on pourrait identifier « consultation » comme table de faits, et par extension les tables de dimensions pourraient être « patient », « diagnostic », « salle » et « mutuelle ». Toutes ces tables de dimensions représentent des catégories de ce qu'une consultation peut contenir en termes de données.

Pour réaliser un modèle conceptuel de donnée efficace est fonctionnelle, il nous faut dans un premier temps imaginer nos dimensions et nos faits. Dans ce but nous avons étudié les fichiers Excel ainsi que la BDD Postgres.

Nous avons donc identifié les dimensions suivantes :

- Patient
- Professionnel Santé
- Diagnostic
- Etablissement Santé
- Salle
- Mutuelle
- Temps Consultation
- Temps Hospitalisation

Concernant les faits :

- Consultation
- Hospitalisation

Une dimension localisation ne nous a pas semblé nécessaire puisque le seul besoin utilisateur lié à la localisation concerne le nombre de décès, donc la table décès.

Or, comme nous le verrons par la suite, il nous semble impossible de faire une jointure entre décès et tous autres tables.

En étudiant la BDD Postgres nous avons remarqué que la table Consultation comporté les clés étrangères de la table Diagnostic, Mutuelle, ProfessionnelSante et patient. Nous avons donc choisi Consultation comme table de faits, et nous avons rajouté en plus des tables de dimensions déjà utilisé la table Salle et TempsConsultation.

Cela nous semble logique puisque qu'une consultation réunit un patient et un professionnel de santé à une certaine date. Une consultation à lieu dans une salle et

peut être couvert par une mutuelle, également une prescription de médicament peut avoir lieu mais nous expliquerons cette gestion dans la partie suivante.

L'objectif est de trouver des dimensions logiques certes, mais aussi qui possèdent des champs permettant de faire des jointures avec la table de fait. Si cela n'est pas possible, lors du mapping, les clés étrangères dans la table de faits seront mélangées. C'est-à-dire que la clé étrangère de patient renverra par exemple un patient qui n'a pas reçu le diagnostic que nous rapporte la clé étrangère de diagnostic. C'est pour cela que des champs permettant de faire des jointures sont importants, ils serviront lorsqu'on alimentera notre table faits, des clés étrangères des tables dimensions.

Ensuite, on retrouve la table de faits Hospitalisation relié aux dimensions Patient, Diagnostic, EtablissementSante et TempsHospitalisation. Comme pour la première table de faits, nous avons pu observer qu'une hospitalisation à une date d'entrée ainsi qu'une durée, ce qui explique la création de la table TempsHospitalisation. Et puis une hospitalisation à lieu dans un établissement de santé pour un patient qui reçoit un diagnostic. Encore une fois, cela a du sens, d'autant plus qu'on retrouve entre chaque table de dimensions une donnée permettant de faire une jointure entre la table de faits et les dimensions énoncées. Soit « identifiant_organisation », présent dans la table de faits et présent dans les dimensions Patient et EtablissementSante. Puis « code_diag » pour diagnostic qu'on retrouve dans la table de faits (voir postgres). Pour la table TempsHospitalisation que nous allons créer, notre mapping (job d'alimentation) permettra de créer la table en faisant correspondre les données liées à la clé primaire d'Hospitalisation avec la clé primaire de TempsHospitalisation que nous créerons.

VII. Modèle de Données

1. Vue Modèle Conceptuel des Données (MCD)

Comme nous l'avons un peu vu dans la partie précédente, notre conception du MCD s'appuie principalement sur l'infrastructure existante de la base de données PostgreSQL et sur les données extraites des fichiers CSV fournis.

Notre MCD va donc suivre nos choix de dimensions et de faits réfléchis dans la partie précédente. Il doit avoir du sens mais aussi être fonctionnelle pour notre partie décisionnelle, comme nous en avons parlé, les jointures sont importantes pour la réalisation des tables de faits. C'est pour cela que l'analyse des champs de chaque tables sources est important pour trouver les potentielles liens entre les entités de notre MCD. Également, il est important de garder en tête l'objectif de base, qui est le besoin de l'utilisateur en termes d'analyse décisionnelle.

C'est donc tous ces points qu'il faut prendre en compte pour réaliser un MCD logique et fonctionnel. Nous allons donc expliciter les différentes parties de notre MCD.

Pour des raisons de concision et de lisibilité dans ce rapport, les schémas ont été condensés pour occuper un espace minimal. Toutefois, pour une exploration approfondie, le fichier complet au format .loo est joint à ce rapport.

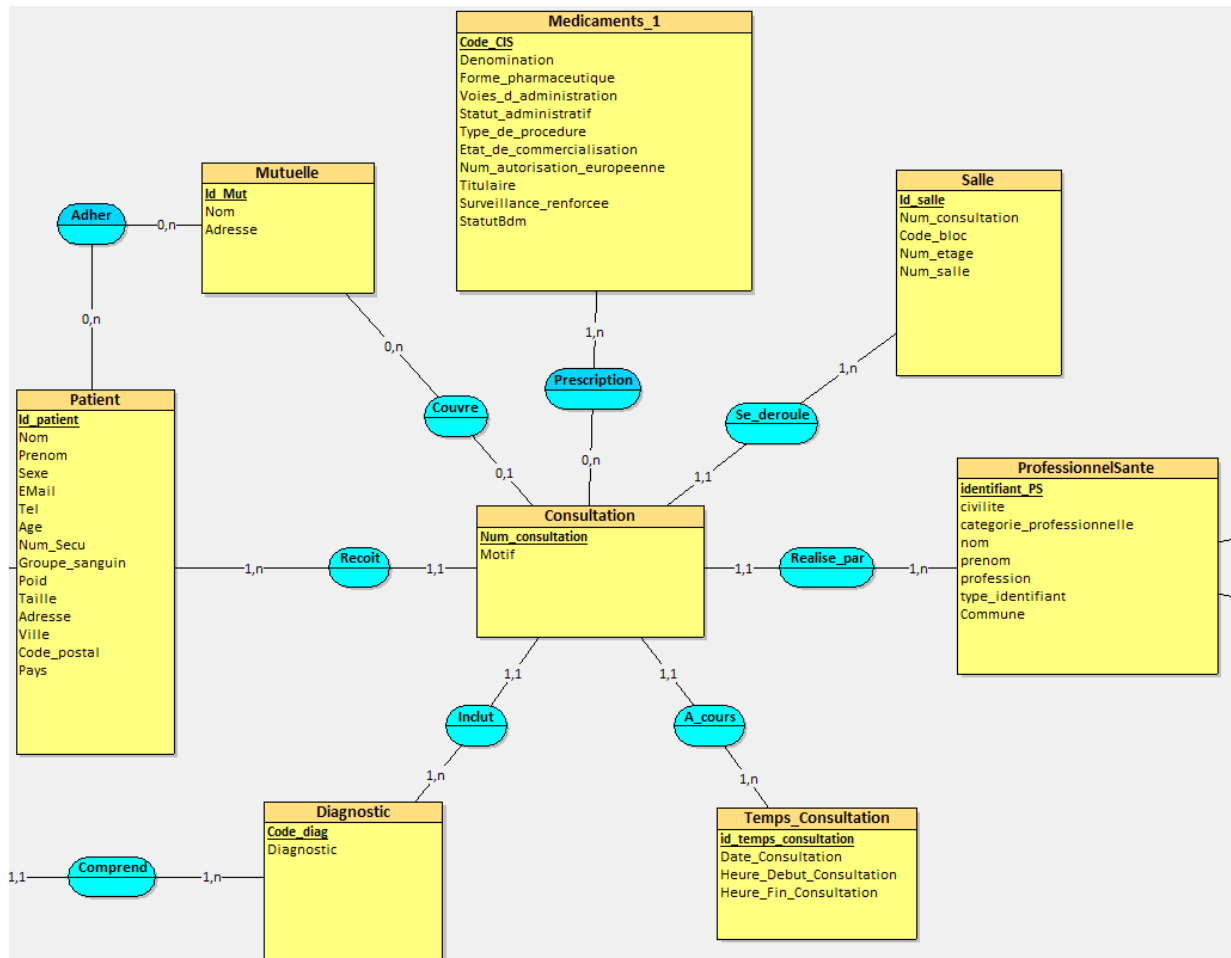


Figure 4 : MCD Etoile Consultation

Ci-dessous la base de notre schéma en étoile sur notre table de faits consultation, l'objectif n'est pas de rentrer dans les détails mais d'en expliquer les grandes lignes.

On y retrouve donc toutes les tables de dimensions citées précédemment, leurs clés primaires se retrouveront donc en clé étrangère dans Consultation, hormis Médicament qui comporte une subtilité comme nous en avons parlé. Nous avons remarqué que dans Postgres se trouvait une table prescription qui se crée grâce aux cardinalités entre Consultation et Médicament, ou se retrouvera alors les clés primaires des deux tables. Cela prend son sens, puisqu'une consultation peut déboucher sur une prescription qui elle comporte des médicaments. Ensuite, on retrouve le lien Adher entre Patient et Mutuelle, qui est le même cas que celui de Prescription. On retrouve la table Adher sur Postgres (que nous verrons par la suite dans le MLD), grâce aux cardinalités des deux tables. Adher comprend donc les patients qui ont souscrit à une mutuelle.

Grâce à ce schéma nous pourrions répondre aux 2 des 8 besoins utilisateurs, soit :

- Taux de consultation des patients par rapport à un diagnostic X sur une période de temps Y
- Taux de consultation par professionnel

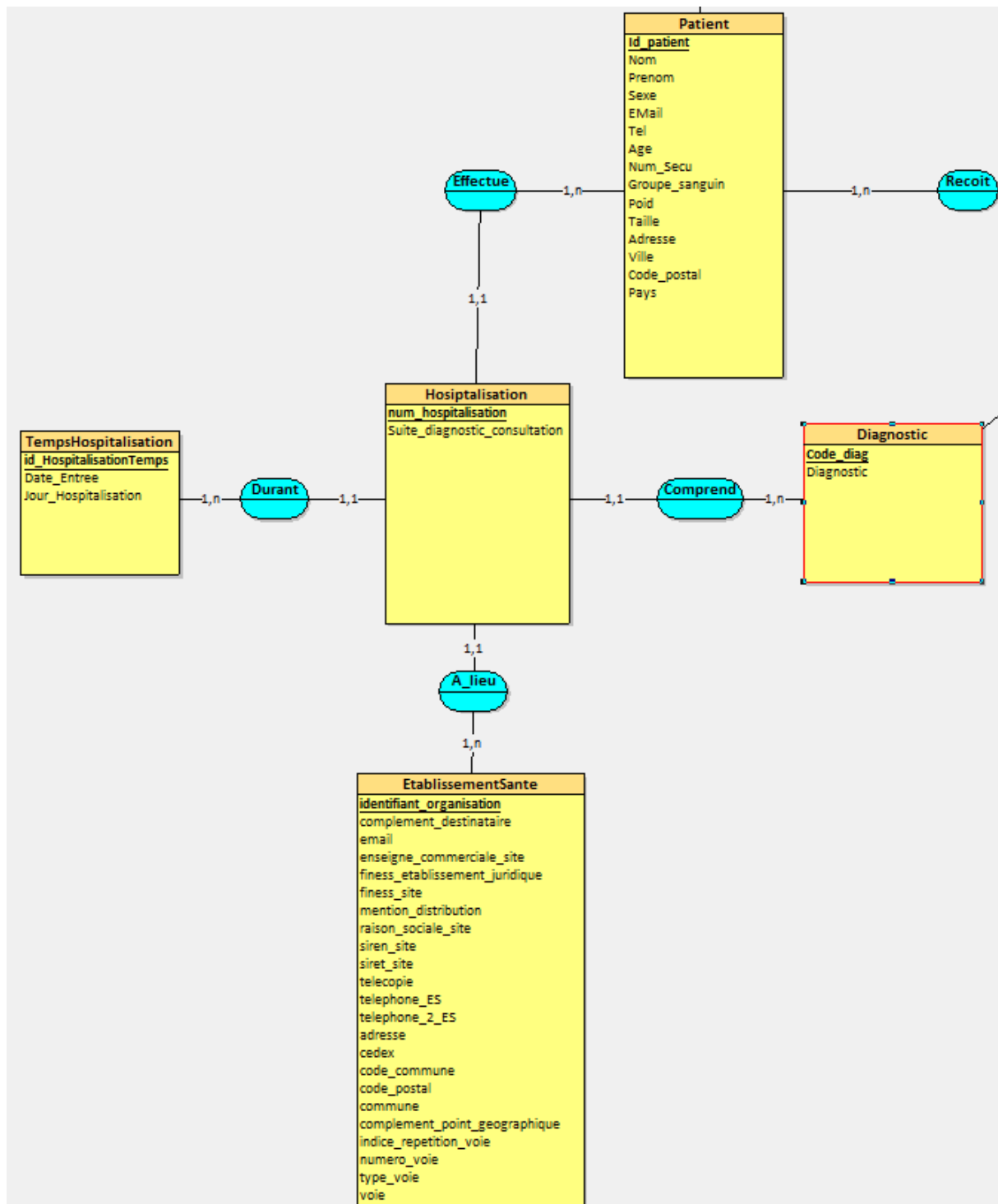


Figure 5 : MCD Etoile Hospitalisation

On retrouve ensuite notre deuxième table de faits Hospitalisation, qui comprends les dimensions TempsHospitalisation, Patient, Diagnostic et EtablissementSante. Le fonctionnement est le même que pour le schéma en étoile précédent, et il nous permettra de répondre à 4 des 8 besoins utilisateurs :

- Taux de consultation des patients dans un établissement X sur une période de temps Y
- Taux global d'hospitalisation des patients dans une période donnée Y
- Taux d'hospitalisation des patients par rapport à des diagnostics sur une période donnée
- Taux d'hospitalisation par sexe, par âge

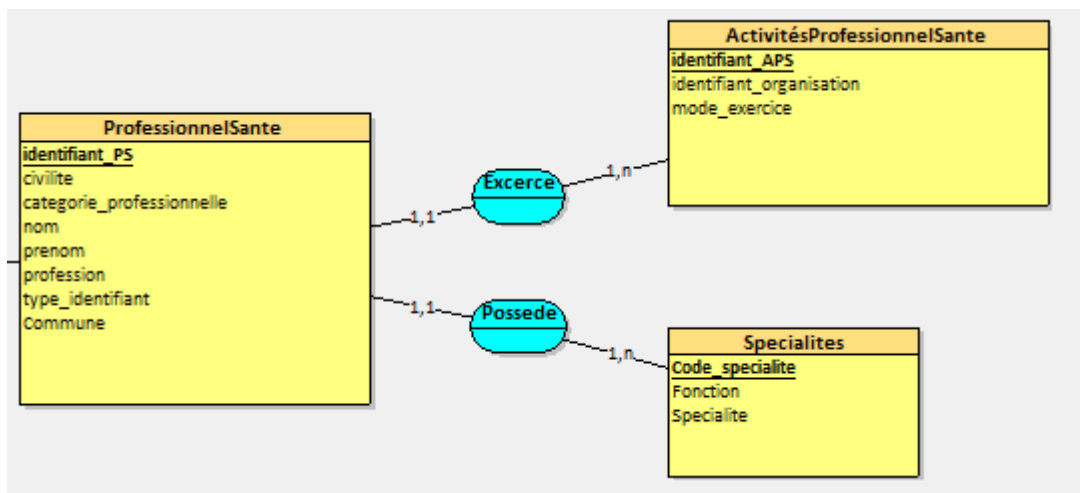


Figure 6 : MCD ProfessionnelSanté

Nous avons deux tables reliées à ProfessionnelSanté, leurs clés primaires se retrouveront dans la table. Et ces deux tables servent à apporter plus de précision sur les informations liées à un professionnel de santé, notamment son activité et sa spécialité.

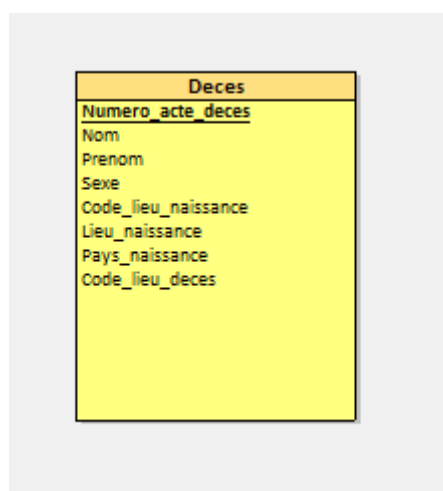


Figure 7 : MCD Décès

Pour finir, on retrouve la table décès qui est isolé puisqu'aucune jointure n'est possible. Elle ne possède aucun champ unique qu'on puisse retrouver dans une autre table, donc si on la lie à la table patiente par exemple, un patient ne pourra pas être relié à un décès car aucun champ ne permet de faire un rapprochement entre un patient et un décès, et cela est le cas pour toutes les autres tables d'après notre analyse.

Cela n'est pas forcément embêtant puisque le besoin utilisateur lié à la table décès est le suivant :

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

Or, on retrouve dans la table décès un « code_lieu_décès » qui pourrait permettre dans la suite du projet de satisfaire ce besoin avec uniquement la table décès. Cela peut évoluer bien entendu.

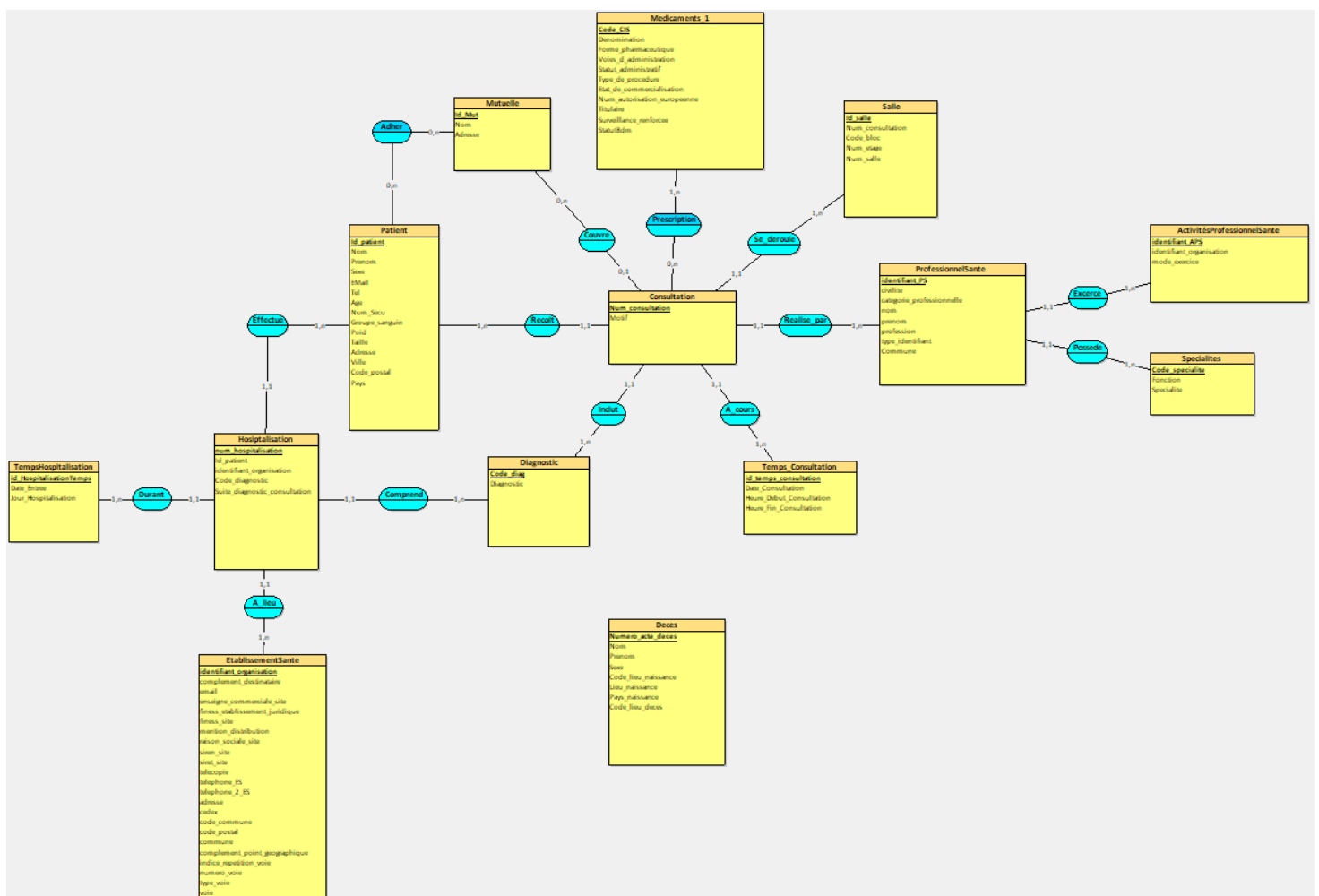


Figure 8 : MCD Global

C'est donc à l'heure d'aujourd'hui ce modèle de donnée qui nous semble le plus pertinent au vu des analyses que nous avons mené. Il peut évidemment amener à être modifié à la suite du projet s'il ne permet pas de répondre aux besoins du client.

Concernant les données liées à la satisfaction, il nous a été demandé de ne pas nous en occuper tout de suite. C'est pour cela que rien n'apparaît dans notre modèle de données à son sujet.

2. Vue Modèle Logique de Données (MLD)

A la suite du MCD nous avons le MLD qui transforme les entités en un schéma relationnel prêt pour l'implémentation. Chaque table contient des champs correspondant aux attributs des entités et est liée par des clés étrangères qui maintiennent l'intégrité relationnelle des données. On retrouve également les tables qui se sont créés par le biais des cardinalités (Adher et Prescription).

La figure ci-dessous illustre le MLD, démontrant la conversion des associations en relations de base de données.

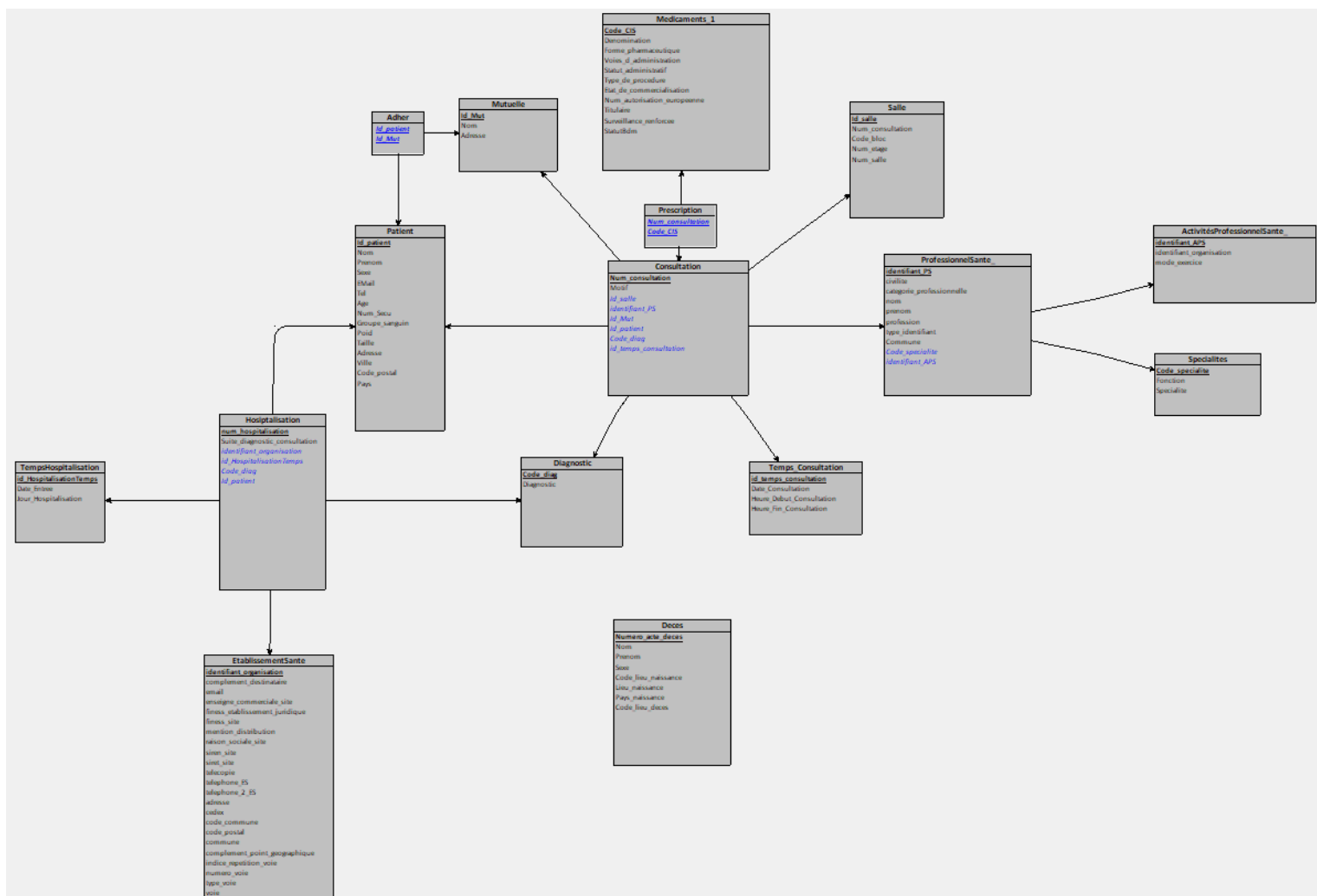


Figure 9 : Modèle Logique de Données

VIII. Job d'alimentation

Comme nous avons pu le voir lors de la présentation de notre architecture décisionnelle, la première étape passe par l'alimentation de notre DLK (DataLake).

Cette alimentation se fait par le biais d'ELT qui ne sont rien d'autre que des jobs Talend, ces derniers vont permettre d'extraire nos données sources pour les intégrer dans notre DLK, c'est-à-dire notre cluster Hadoop sous Cloudera.

L'objectif du DLK est de stocker des données brutes, qui n'ont subi aucune transformation. C'est pour cela que nous verrons que nos jobs ne font qu'extraire des données non structurées (donnée stockée dans son format d'origine et non traitée avant son utilisation) de nos sources pour alimenter le DLK. On ne retrouve que des jointures qui servent à créer nos tables faites, afin que les clés étrangères correspondent entre elles. Avec pour objectif de créer des fichiers .txt fonctionnels des tables pour la partie traitement ETL, afin d'alimenter le DWH (DataWare House).

Voici un exemple du job qui nous permet d'alimenter la table Hospitalisation, et qui alimente également la table TempsHospitalisation.

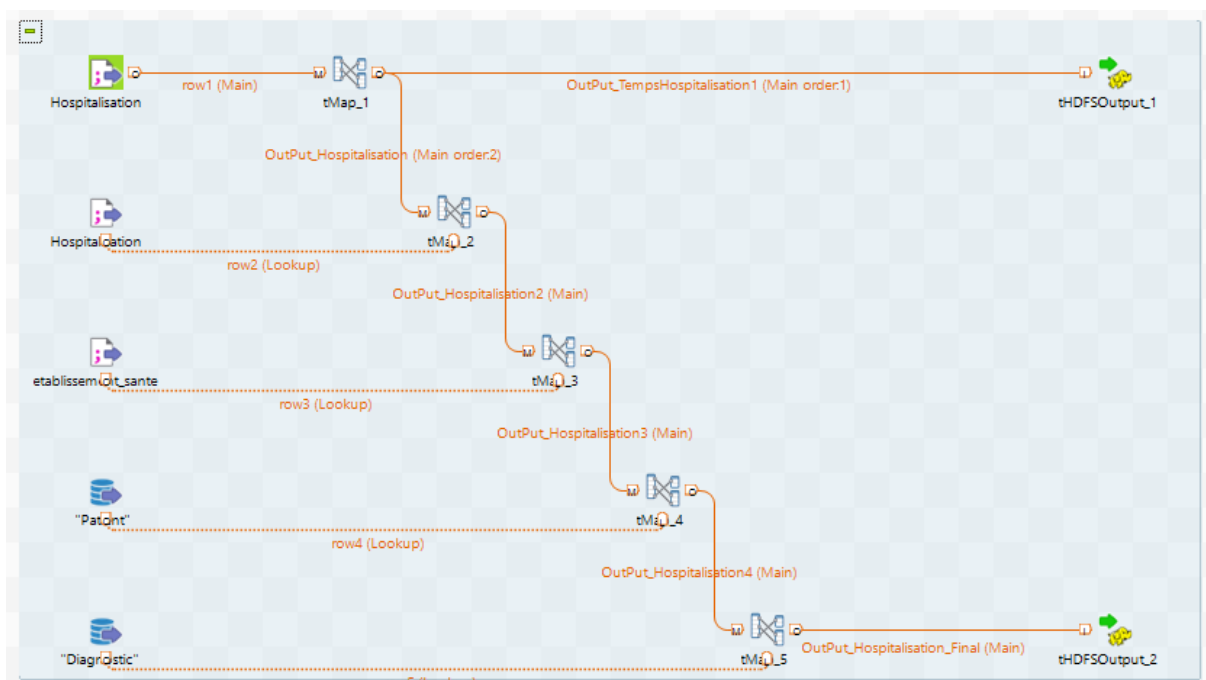


Figure 10 : Exemple Job d'Alimentation

Lors du premier tMap nous alimentons le OutPut de TempsHospitalisation en créant l'id de la table de cette façon :

`Numeric.sequence("TempsHospitalisation", 1, 1)` 🔑 `id_HospitalisationTemps`

On reprend également cette id qu'on insère en tant que clé étrangère dans la table Hospitalisation, ensuite chaque tMap réalise une jointure pour insérer la clé étrangère de chaque dimension. Ci-dessous un exemple avec EtablissementSante :

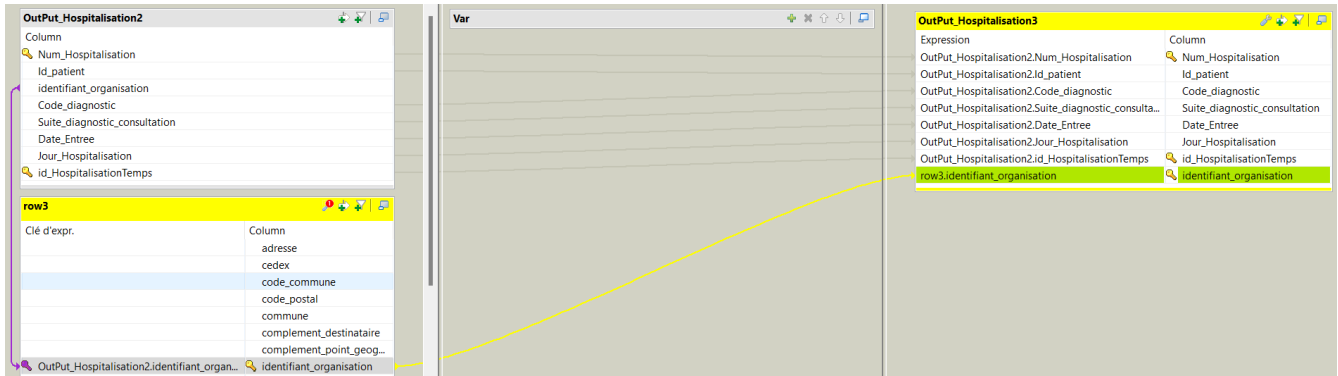


Figure 11 : tMap EtablissementSante

On réalise une jointure sur l'identifiant organisation commune aux deux tables pour rajouter la clé étrangère à la table Hospitalisation. Une fois la table alimentée on la charge dans HDFS, notre cluster sur Cloudera.

On retrouve de schéma de jointure lorsqu'on a besoin de faire passer une clé étrangère sur une autre table. Les clés étrangères permettront de réaliser des requêtes entres différentes tables pour répondre aux besoins utilisateurs.

Pour les alimentations de table sans clé étrangère on retrouve un schéma d'alimentation classique, comme pour la table Patient :

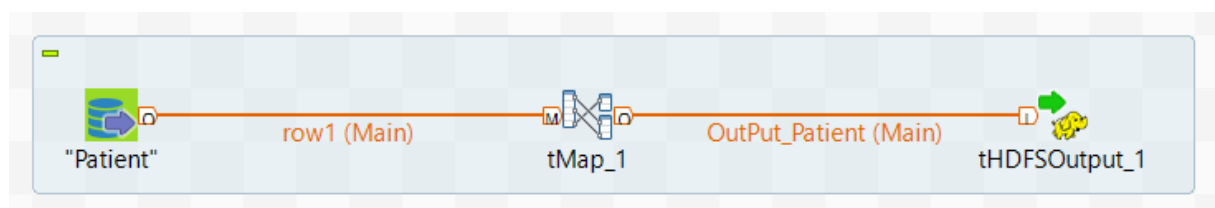


Figure 12 : Job Patient

Nous avons fait le choix de réaliser les jobs d'alimentation dans plusieurs jobs distincts plutôt que tout regrouper dans un seul job.

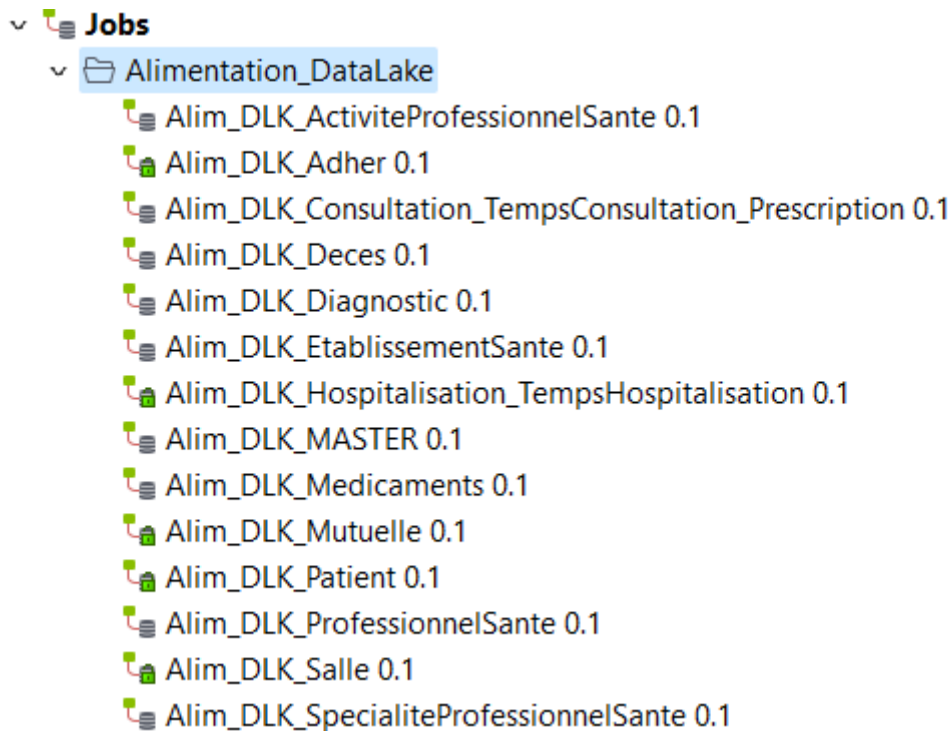


Figure 13 : Ensemble des jobs d'alimentation

En divisant les processus d'alimentation en plusieurs jobs modulaires, chaque job peut se concentrer sur une tâche spécifique ou une source de données particulière. Cela rend les jobs plus faciles à comprendre, à maintenir et à réutiliser dans différents scénarios.

La modularité permet également une meilleure gestion des changements. Si une partie du processus doit être modifiée, il est plus facile de localiser et de mettre à jour le job pertinent sans perturber les autres parties du processus.

Cependant, le problème de cette démarche impose de lancer les jobs un par un. Pour résoudre ce problème nous avons créé un job « MASTER » ce dernier comporte des tRunJob qui pointe vers chacun des autres jobs.

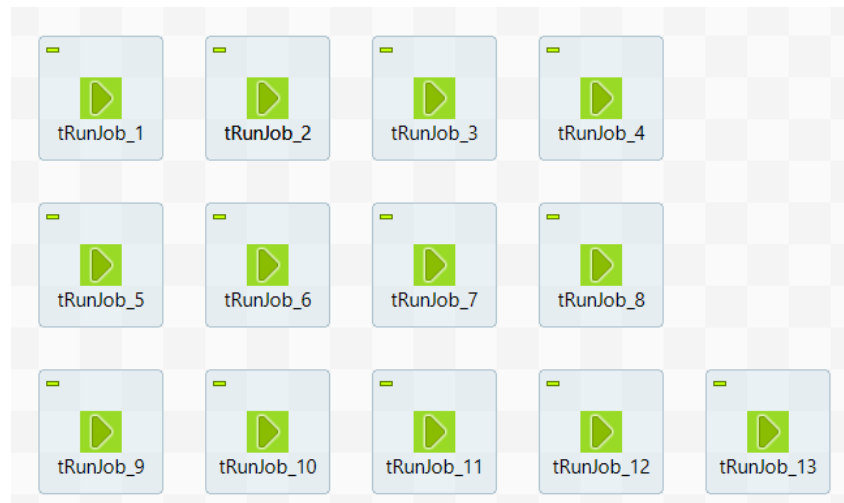


Figure 14 : Job Master

Il suffit donc d'exécuter ce job master pour lancer tous les traitements d'alimentation du Datalake. En exécutant les jobs en parallèle on peut réduire considérablement le temps nécessaire pour traiter l'ensemble des données. Plutôt que d'attendre la fin de chaque job avant de commencer le suivant, les tâches peuvent être exécutées simultanément, ce qui permet d'achever le traitement plus rapidement. Le screen ci-dessus montre la bonne exécution des jobs avec les lignes de données qui sont chargées dans le DLK. Pour vérifier si l'alimentation a été réussie, il suffit de se rendre sur HUE qui est une interface web/utilisateur pour Hadoop. Et l'on peut constater le chargement des données dans notre DLK :

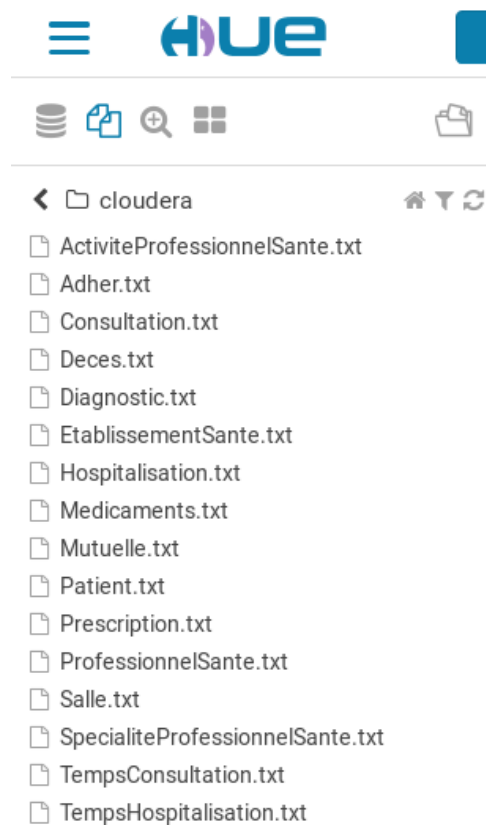


Figure 15 : Fichiers d’Alimentation du Datalake

IX. Automatisation des jobs

Une architecture décisionnelle efficiente implique l'instauration d'un système d'automatisation, clé pour la rationalisation du temps et la sécurisation des processus. L'automatisation est conçue pour confier les tâches récurrentes à des systèmes automatisés, permettant ainsi aux équipes de se focaliser sur des initiatives stratégiques et d'innovation. Cette rationalisation des processus se traduit par une réduction conséquente des coûts et une diminution des erreurs, qui sont souvent coûteuses à rectifier.

Concernant l'alimentation de notre entrepôt de données, nous avons opté pour une approche par lots (batch), nécessitant une fréquence d'alimentation mûrement réfléchie. Pour garantir l'intégrité des données, nous avons choisi une fréquence uniforme pour toutes les tables, plutôt que de se concentrer uniquement sur certaines d'entre elles. Cette uniformité assure la cohérence globale de nos données et prévient les décalages qui pourraient survenir en cas de mises à jour incohérentes.

Nous avons déterminé une fréquence d'alimentation mensuelle, alignée sur les besoins exprimés par les utilisateurs, tout en restant ouverts à des ajustements ultérieurs en fonction de l'évolution des exigences opérationnelles. Cette flexibilité nous permet de

rester en mesure de basculer vers un traitement en temps réel si les données requièrent une analyse instantanée pour en tirer des insights en direct.


Pour mettre en œuvre cette automatisation, nous avons testé Apache Airflow déployé dans un conteneur Docker, mais nous avons rencontré des difficultés liées aux chemins de fichiers sources situés sur l'hôte Windows. Jenkins était une autre alternative, mais nous avons finalement privilégié un mécanisme direct et sans intermédiaire.

Ainsi, nous avons décidé de construire le job "Master" avec Talend, qui pilotera l'exécution des autres jobs d'alimentation. L'exportation de ce job "Master" génère un fichier script.bat, que nous prévoyons de déclencher automatiquement via le Planificateur de tâches de Windows, chaque premier jour du mois. Cette stratégie simplifie notre processus d'automatisation et réduit la complexité opérationnelle, tout en bénéficiant des avantages de l'automatisation.

Notre démarche associe donc efficience et simplicité, assurant une intégration transparente et une gestion maîtrisée des coûts, tout en demeurant flexible pour adopter des solutions d'automatisation plus sophistiquées si les besoins métier le justifient à l'avenir.

Figure 16 : Planificateur de tâches

Assistant Créer une tâche de base ×

 Démarrer un programme

Créer une tâche de base

Déclencheur	Programme/script :	
Une fois	C:\installation_BigData\Automatisation_job\Alim_DLK_MASTER_0.1\Alim_	<button>Parcourir...</button>
Action	Ajouter des arguments (facultatif) :	<input type="text"/>
Démarrer un programme	Commencer dans (facultatif) :	<input type="text"/>
Terminer		

Figure 17 : Script job master.bat

X. Conclusion

Pour conclure, nous avons réalisé à travers ce livrable une proposition d'architecture décisionnelle. Cette dernière a été pensée afin d'optimiser la gestion et l'analyse des données au sein du groupe CHU. Dans cette proposition, nous avons intégré différents outils et technologies que nous avons à disposition, comme Hadoop et Talend.

Afin de créer notre modèle de donnée, nous avons établis des dimensions et des faits afin de répondre du mieux possible aux besoins des utilisateurs. Ce travail de préparation nous a facilité la réalisation de notre Modèle Conceptuel de Donnée, permettant de construire un modèle pertinent et fonctionnel.

A la suite de ces tâches nous avons commencé la première étape de notre architecture décisionnelle, comportant l'extraction des données sources et leurs chargements dans le Datalake. Pour réaliser cette alimentation, nous avons mis en place des jobs Talend en traitement batch, c'est-à-dire automatisé à une fréquence mensuelle.

La prochaine étape consistera au traitement des données par ETL pour charger le DWH, avec pour objectif final d'utiliser les données traitées pour de la visualisation de données.

Webographie

<https://medium.com/helpshift-engineering/building-a-data-warehouse-with-hive-at-helpshift-part-1-443046df6484>

<https://www.analyticsvidhya.com/blog/2021/05/hive-a-data-warehouse-in-hadoop-framework/>

<https://www.lebigdata.fr/apache-hive-definition>

<https://data-flair.training/blogs/apache-hive-architecture/>

<https://meritis.fr/hive-et-big-data-exploration-des-concepts/>

<https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql>

<https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4462471-maitrisez-le-theoreme-de-cap>

<https://infodecisionnel.com/data-management/big-data/acid-versus-base/>

https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.1.0/using-hiveql/content/hive_using_materialized_views.html

<https://www.talend.com/fr/resources/elt-vs-etl/>

<https://www.ediservices.com/fr/etl-integration/>

<https://www.cartelis.com/blog/data-lake-vs-data-warehouse/>

<https://openclassrooms.com/fr/courses/4467481-creez-votre-data-lake/4467488-identifiez-les-besoins-de-votre-data-lake>

<https://www.data-transitionnumerique.com/cube-olap-decisionnel-big-data/>

https://moodle.cesi.fr/pluginfile.php/24914/mod_resource/content/4/res/Informatique_Desicionnelle.pdf

<https://www.edureka.co/blog/videos/etl-using-big-data-talend/>

<https://www.edureka.co/blog/talend-big-data-tutorial/>

<https://blog.octo.com/levolution-des-architectures-decisionnelles-avec-big-data/>

<https://www.cetic.be/Comment-deployer-avec-succes-un-projet-Big-Data>

<https://www.solution-bi.com/fr/blog/la-modernisation-du-data-warehouse-a-leredubig-data>

<https://docs.microsoft.com/fr-fr/azure/architecture/guide/architecture-styles/big-data>

<https://docs.microsoft.com/fr-fr/power-bi/fundamentals/power-bi-overview>