



PROJET CLOUD HEALTHCARE UNIT

Livrable 2

Résumé

Cette documentation consiste en l'illustration des requêtes analytiques mises en place dans le cadre d'une transformation digitale majeure initiée par le groupe CHU, ayant pour objectif d'exploiter la quantité considérable de données générées par les systèmes de gestion de soins

GROUPE N°4 CAVARROC, FEVRE, ZAGO

Table des matières

I.	Contexte	1
II.	Modèle physique	2
	Dimension Patient.....	3
	Dimension Professionnel	4
	Dimension Temps	5
	Dimension Localisation	6
	Dimension Diagnostic	6
	Peuplement des différentes tables.....	7
	Faits.....	9
III.	Requêtes d'analyse	11
	Taux de consultation des patients dans un établissement sur une période	11
	Taux de consultation des patients par rapport à un diagnostic sur une période	12
	Taux global d'hospitalisation des patients dans une période donnée	12
	Taux d'hospitalisation des patients par rapport à des diagnostics sur une période donnée	13
	Taux d'hospitalisation/consultation par sexe, par âge	14
	Taux de consultation par professionnel	15
	Nombre de décès par localisation (région) et sur l'année 2019.....	16
	Taux global de satisfaction par région sur l'année 2020	16
IV.	Evaluation de performance	18
V.	Conclusion	20
VI.	Annexe	21

Table des Illustrations

Figure 1 Sortie de la tâche d'intégration de la dimension Localisation.....	2
Figure 2 Requête de création de la base de données	2
Figure 3 Création de la table non partitionnée Patient.....	3
Figure 4 Création de la table Patient	3
Figure 5 Création de la table non partitionnée Professionnel	4
Figure 6 Création de la table Professionnel.....	4
Figure 7 Création de la table non partitionnée Temps.....	5
Figure 8 Création de la table Temps	5
Figure 9 Création de la table non partitionnée Localisation	6
Figure 10 Création de la table Localisation.....	6
Figure 11 Création de la table Diagnostic.....	7
Figure 12 Instruction LOCATION de la dimension Localisation	7
Figure 13 Requête de peuplement de la dimension Patient.....	7
Figure 15 Ensemble des répertoires à propos du projet présents dans l'HDFS	8
Figure 14 Extrait du fichier texte Localisation présent dans l'HDFS.....	8
Figure 16 Création de la table de fait	9
Figure 17 Extrait du fichier texte fait présent dans l'HDFS.....	9
Figure 18 Création de la vue du taux de consultation par établissement.....	11
Figure 19 Création de la vue du taux de consultation par diagnostic	12
Figure 20 Création de la vue du taux d'hospitalisation globale.....	13
Figure 21 Création de la vue du taux d'hospitalisation par diagnostic.....	14
Figure 22 Création des vues des taux d'hospitalisation et de consultation par âge et sexe...	15
Figure 23 Création de la vue du taux de consultation par professionnel.....	15
Figure 24 Création du la vue du nombre de morts par région	16
Figure 25 Création de la vue de satisfaction par région	16
Figure 26 Résultats des tests de temps d'exécutions des requêtes analytiques	18
Figure 27 Résultats des tests de temps d'exécutions d'une requête simplifiée	19
Figure 28 - Table des diagnostics	21
Figure 29 - Table Patient	21
Figure 30 - Table Professionnel.....	21
Figure 31 - Table Temps.....	22
Figure 32 - Table Localisation	22
Figure 33 - Table des Faits	22

I. Contexte

Le Groupe CHU (Cloud Healthcare Unit) a reconnu l'intérêt et la nécessité d'une transformation numérique majeure. Il a été demandé à notre département de l'aider à construire son propre entrepôt de données, ce qui permettrait au groupe de tirer parti des grandes quantités de données générées par le système de gestion des soins.

L'objectif est de pouvoir répondre à une variété de besoins d'accès et d'analyse des utilisateurs. À cette fin, le Groupe CHU attend :

- Une solution complète de modèles, d'outils et d'architecture pour extraire et stocker des données, qui peuvent ensuite être explorées et visualisées selon différents critères.
- Une solution pour intégrer les données de fichiers distribués dans une seule source persistante.
- Identifier les besoins des utilisateurs (praticiens, responsables d'établissements) en analyse de données pour une utilisation directe dans le suivi des patients au niveau national et à long terme
- Des conseils sur les outils d'intégration, les logiciels de stockage et de visualisation adaptés, et l'exploration de données entièrement sécurisée pour faciliter une meilleure prise de décision.

Cette étude tente de relever tous ces défis dans le secteur de la santé. Par conséquent, des solutions doivent être développées pour répondre aux exigences d'infrastructure des données médicales, telles que le rapport coût/efficacité, la sécurité, la résilience et l'évolutivité.

Les grandes phases envisagées du projet sont les suivantes :

- Etude de l'architecture à mettre en place
- Modélisation conceptuelle des données
- Implémentation physique des données
- Exploitation, mesure de performance, optimisations.

Ce deuxième livrable comprend l'implémentation physique des données et leur exploitation, ainsi que les mesures de performances et leur optimisation.

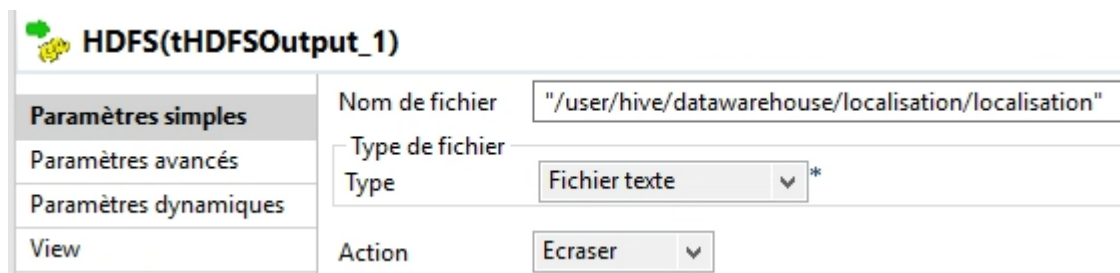
II. Modèle physique

Après l'exécution tous les jobs d'alimentation, dont les détails d'exécution ont été évoqués dans le précédent livrable, les fichiers textes en sortie sont stockés sur Hive, dans le répertoire « */user/hive/datawarehouse* ».

Chaque job d'alimentation a un chemin de fichier correspondant au nom de sa dimension, comme suit :

/user/hive/datawarehouse/<DossierDuNomDeLaDimension>/<FichierDuNomDeLaDimension>

Ce type de répertoire (un dossier par dimension) facilite la création de tables dans Hive. Voici un exemple de chemin de fichier de sortie pour le job de la dimension *Localisation* :



The screenshot shows a web interface for configuring HDFS output. On the left, there is a sidebar with tabs: 'Paramètres simples' (selected), 'Paramètres avancés', 'Paramètres dynamiques', and 'View'. The main area is titled 'HDFS(tHDFSOutput_1)'. It contains three input fields: 'Nom de fichier' with the value '/user/hive/datawarehouse/localisation/localisation', 'Type de fichier' with a dropdown menu showing 'Fichier texte' and an asterisk, and 'Action' with a dropdown menu showing 'Ecraser'.

Figure 1 Sortie de la tâche d'intégration de la dimension Localisation

Nous avons créé une base de données appelée CHU pour pouvoir y stocker les différentes tables permettant l'analyse des données. Voici la requête :

```
CREATE DATABASE IF NOT EXISTS CHU
COMMENT 'BDD du projet CHU'
LOCATION '/user/hive/datawarehouse';
```

Figure 2 Requête de création de la base de données

Ensuite, pour chaque dimension qui nécessite un partitionnement, une autre table « temporaire » est créée, de manière à pouvoir comparer la durée d'exécution entre une table avec partitionnement et sans. Les dimensions qui nécessitent un partitionnement sont la dimension patient, professionnel, localisation et temps.

Dimension Patient

Voici la requête de création de la table temporaire (autrement dit sans partitionnement) de la dimension *patient* :

```
CREATE EXTERNAL TABLE IF NOT EXISTS Patient_tmp(  
  Id int,  
  Id_patient int,  
  Sexe_patient char(1),  
  Tranche_age_patient int,  
  Jour_naissance_patient int,  
  Mois_naissance_patient int,  
  Annee_naissance_patient int)  
COMMENT 'Dimension patient non partitionne'  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\';'  
STORED AS TEXTFILE  
LOCATION '/user/hive/datawarehouse/patient';
```

Figure 3 Création de la table non partitionnée Patient

La commande *LOCATION* permet de récupérer les données du job d'alimentation, et prend un chemin de fichier en paramètre. Cette première table permet la création de la deuxième, avec cette fois-ci un partitionnement :

```
CREATE EXTERNAL TABLE IF NOT EXISTS Patient(  
  Id int,  
  Id_patient int,  
  Jour_naissance_patient int,  
  Mois_naissance_patient int,  
  Annee_naissance_patient int)  
COMMENT 'Dimension patient'  
PARTITIONED BY (Sexe_patient char(1), Tranche_age_patient int)  
CLUSTERED BY (Id) into 10 BUCKETS  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\';'  
STORED AS TEXTFILE;  
  
INSERT OVERWRITE TABLE Patient PARTITION (Sexe_patient, Tranche_age_patient)  
SELECT Id, Id_patient, Jour_naissance_patient, Mois_naissance_patient, Annee_naissance_patient, Sexe_patient,  
  Tranche_age_patient FROM Patient_tmp;
```

Figure 4 Création de la table Patient

La première commande permet de créer la table *patient* sans la remplir. La seconde récupère les données de la table temporaire *patient_tmp* pour remplir la table *patient*.

Cette table est partitionnée en fonction du sexe, puis de la tranche d'âge du patient. De plus, les données partitionnées sont stockées selon une technique d'organisation des données en parties plus petites, les *buckets*. Nous avons opté pour dix *buckets*, en fonction de l'identifiant de la ligne dans la table afin de répartir l'ensemble des données de manière uniforme.

Dimension Professionnel

Voici la requête de création de la table temporaire (autrement dit sans partitionnement) de la dimension *Professionnel* :

```
CREATE EXTERNAL TABLE IF NOT EXISTS Professionnel_tmp(  
  Id int,  
  Identifiant_professionnel string,  
  Nom_professionnel string,  
  Prenom_professionnel string,  
  Identifiant_etablissement_professionnel string,  
  Libelle_etablissement_professionnel string,  
  Code_departement_etablissement_professionnel string,  
  Profession_professionnel string,  
  Specialite_professionnel string,  
  Mode_exercice_professionnel string)  
COMMENT 'Dimension professionnel non partionne'  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\\;'  
STORED AS TEXTFILE  
LOCATION '/user/hive/datawarehouse/professionnel';
```

Figure 5 Création de la table non partitionnée *Professionnel*

La commande *LOCATION* permet de récupérer les données du job d'alimentation, et prend un chemin de fichier en paramètre. Cette première table permet la création de la deuxième, avec cette fois-ci un partitionnement :

```
CREATE EXTERNAL TABLE IF NOT EXISTS Professionnel(  
  Id int,  
  Identifiant_professionnel string,  
  Nom_professionnel string,  
  Prenom_professionnel string,  
  Identifiant_etablissement_professionnel string,  
  Libelle_etablissement_professionnel string,  
  Code_departement_etablissement_professionnel string,  
  Specialite_professionnel string,  
  Mode_exercice_professionnel string)  
COMMENT 'Dimension professionnel'  
PARTITIONED BY (Profession_professionnel string)  
CLUSTERED BY (Id) into 100 BUCKETS  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\\;'  
STORED AS TEXTFILE;  
  
INSERT OVERWRITE TABLE Professionnel PARTITION (Profession_professionnel)  
SELECT Id, Identifiant_professionnel, Nom_professionnel, Prenom_professionnel,  
  Identifiant_etablissement_professionnel, Libelle_etablissement_professionnel,  
  Code_departement_etablissement_professionnel, Specialite_professionnel, Mode_exercice_professionnel,  
  Profession_professionnel FROM Professionnel_tmp;
```

Figure 6 Création de la table *Professionnel*

La première commande permet de créer la table *Professionnel* sans la remplir. La seconde récupère les données de la table temporaire *professionnel_tmp* pour remplir la table *Professionnel*.

Cette table est partitionnée en fonction de la profession. De plus, les données partitionnées sont stockées selon une technique d'organisation des données en parties plus petites, les *buckets*. Nous avons opté pour cent buckets, en fonction de l'identifiant de la ligne dans la table.

Dimension Temps

Voici la requête de création de la table temporaire (autrement dit sans partitionnement) de la dimension temps :

```
CREATE EXTERNAL TABLE IF NOT EXISTS Temps_tmp(  
  Id int,  
  Jour_temps int,  
  Mois_temps int,  
  Annee_temps int)  
COMMENT 'Dimension temps non partitionne'  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\\';  
STORED AS TEXTFILE  
LOCATION '/user/hive/datawarehouse/temps';
```

Figure 7 Création de la table non partitionnée Temps

La commande *LOCATION* permet de récupérer les données du job d'alimentation, et prend un chemin de fichier en paramètre. Cette première table permet la création de la deuxième, avec cette fois-ci un partitionnement :

```
CREATE EXTERNAL TABLE IF NOT EXISTS Temps(  
  Id int,  
  Jour_temps int,  
  Mois_temps int)  
COMMENT 'Dimension temps'  
PARTITIONED BY (Annee_temps int)  
CLUSTERED BY (Id) into 5 BUCKETS  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\\';  
STORED AS TEXTFILE;  
  
INSERT OVERWRITE TABLE Temps PARTITION (Annee_temps)  
SELECT Id, Jour_temps, Mois_temps, Annee_temps FROM Temps_tmp;
```

Figure 8 Création de la table Temps

La première commande permet de créer la table *Temps* sans la remplir. La seconde récupère les données de la table temporaire *temps_tmp* pour remplir la table *Temps*.

Cette table est partitionnée en fonction de l'année. De plus, les données partitionnées sont stockées selon une technique d'organisation des données en parties plus petites, les *buckets*. Nous avons opté pour cinq buckets, en fonction de l'identifiant de la ligne dans la table.

Dimension Localisation

Voici la requête de création de la table temporaire (autrement dit sans partitionnement) de la dimension *Localisation* :

```
CREATE EXTERNAL TABLE IF NOT EXISTS Localisation_tmp(  
  Id int,  
  Code_departement_localisation string,  
  Libelle_departement_localisation string,  
  Code_region_localisation string,  
  Libelle_region_localisation string)  
COMMENT 'Dimension localisation non partitionne'  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\';'  
STORED AS TEXTFILE  
LOCATION '/user/hive/datawarehouse/localisation';
```

Figure 9 Création de la table non partitionnée Localisation

La commande *LOCATION* permet de récupérer les données du job d'alimentation, et prend un chemin de fichier en paramètre. Cette première table permet la création de la deuxième, avec cette fois-ci un partitionnement :

```
CREATE EXTERNAL TABLE IF NOT EXISTS Localisation(  
  Id int,  
  Code_departement_localisation string,  
  Libelle_departement_localisation string,  
  Libelle_region_localisation string)  
COMMENT 'Dimension diagnostic'  
PARTITIONED BY (Code_region_localisation string)  
CLUSTERED BY (Id) into 2 BUCKETS  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\';'  
STORED AS TEXTFILE;  
  
INSERT OVERWRITE TABLE Localisation PARTITION (Code_region_localisation)  
SELECT Id, Code_departement_localisation, Libelle_departement_localisation, Libelle_region_localisation,  
Code_region_localisation FROM Localisation_tmp;
```

Figure 10 Création de la table Localisation

La première commande permet de créer la table localisation sans la remplir. La seconde récupère les données de la table temporaire *localisation_tmp* pour remplir la table *localisation*.

La table *Localisation* est partitionnée en fonction du code de la région. De plus, les données partitionnées sont stockées selon une technique d'organisation des données en parties plus petites, les *buckets*. Nous avons opté pour deux buckets, en fonction de l'identifiant de la ligne dans la table.

Dimension Diagnostic

La table *Diagnostic* n'a pas été partitionnée, étant donné que les données ne sont pas redondantes : une ligne de la table correspond à un diagnostic précis et son code correspondant.

Voici la requête de création de la table :

```
CREATE EXTERNAL TABLE IF NOT EXISTS Diagnostic(  
  Id int,  
  Code_diagnostic string,  
  Libelle_diagnostic string)  
COMMENT 'Dimension diagnostic'  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\';'  
STORED AS TEXTFILE  
LOCATION '/user/hive/datawarehouse/diagnostic';
```

Figure 11 Création de la table Diagnostic

Peuplement des différentes tables

Pour ce qui est du peuplement des différentes dimensions on peut observer deux méthodes au sein des différentes captures d'écran :

- La première méthode est utilisée pour peupler les tables non partitionnées ce qui représentent ici la première étape de création de chacune des tables. Cette méthode repose sur l'instruction *LOCATION* qui permet à une table de cibler un dossier afin de se servir des données en son sein pour peupler ses lignes. Voici un exemple de l'instruction *LOCATION* :

```
LOCATION '/user/hive/datawarehouse/localisation';
```

Figure 12 Instruction LOCATION de la dimension Localisation

- Pour la deuxième méthode, il est cette fois-ci question de remplir une table à l'aide du contenu d'une autre préalablement peuplée. Pour réaliser cette tâche, nous nous sommes servis de l'instruction *INSERT* détaillée ci-dessous et déjà présente dans la capture d'écran plus au-dessus :

```
INSERT OVERWRITE TABLE Patient PARTITION (Sexe_patient, Tranche_age_patient)  
SELECT Id, Id_patient, Jour_naissance_patient, Mois_naissance_patient,  
Annee_naissance_patient, Sexe_patient, Tranche_age_patient FROM Patient_tmp;
```

Figure 13 Requête de peuplement de la dimension Patient

Ainsi, peu importe la méthode appliquée, l'ensemble du peuplement repose sur la bonne exécution des job Talend. Effectivement, si les fichiers HDFS ne sont pas présents dans le répertoire cible alors aucune des dimensions ne sera peuplée. Cependant dans notre cas et comme confirmé dans le premier livrable, l'ensemble des jobs d'intégration de données sont concluants et nous permettrons donc de compléter l'intégralité du modèle sous HIVE afin de pouvoir requêter sur nos données. Après exécution des jobs, on retrouve effectivement l'ensemble des fichiers texte HDFS correspondant au sein du bon répertoire.

```

[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/datawarehouse
Found 6 items
drwxrwxrwx - cloudera supergroup 0 2022-05-17 09:00 /user/hive/datawarehouse/diagnostic
drwxrwxrwx - cloudera supergroup 0 2022-05-17 15:43 /user/hive/datawarehouse/fait
drwxrwxrwx - cloudera supergroup 0 2022-05-17 23:10 /user/hive/datawarehouse/localisation
drwxrwxrwx - cloudera supergroup 0 2022-05-17 12:58 /user/hive/datawarehouse/patient
drwxrwxrwx - cloudera supergroup 0 2022-05-17 09:19 /user/hive/datawarehouse/professionnel
drwxrwxrwx - cloudera supergroup 0 2022-05-17 09:22 /user/hive/datawarehouse/temps
[cloudera@quickstart ~]$ █

```

Figure 15 Ensemble des répertoires à propos du projet présents dans l'HDFS

```

[cloudera@quickstart ~]$ hdfs dfs -cat /user/hive/datawarehouse/localisation/localisation
1;01;Ain;84;Auvergne-Rhône-Alpes
2;02;Aisne;32;Hauts-de-France
3;03;Allier;84;Auvergne-Rhône-Alpes
4;04;Alpes-de-Haute-Provence;93;Provence-Alpes-Côte d'Azur
5;05;Hautes-Alpes;93;Provence-Alpes-Côte d'Azur
6;06;Alpes-Maritimes;93;Provence-Alpes-Côte d'Azur
7;07;Ardèche;84;Auvergne-Rhône-Alpes
8;08;Ardennes;44;Grand Est
9;09;Ariège;76;Occitanie
10;10;Aube;44;Grand Est
11;11;Aude;76;Occitanie
12;12;Aveyron;76;Occitanie
13;13;Bouches-du-Rhône;93;Provence-Alpes-Côte d'Azur
14;14;Calvados;28;Normandie
15;15;Cantal;84;Auvergne-Rhône-Alpes
16;16;Charente;75;Nouvelle-Aquitaine
17;17;Charente-Maritime;75;Nouvelle-Aquitaine
18;18;Cher;24;Centre-Val de Loire
19;19;Corrèze;75;Nouvelle-Aquitaine
20;20;Corse;94;Corse
21;21;Côte-d'Or;27;Bourgogne-Franche-Comté
22;22;Côtes-d'Armor;53;Bretagne
23;23;Creuse;75;Nouvelle-Aquitaine
24;24;Dordogne;75;Nouvelle-Aquitaine
25;25;Doubs;27;Bourgogne-Franche-Comté
26;26;Drôme;84;Auvergne-Rhône-Alpes
27;27;Eure;28;Normandie
28;28;Eure-et-Loir;24;Centre-Val de Loire
29;29;Finistère;53;Bretagne
30;2A;Corse-du-Sud;94;Corse
31;2B;Haute-Corse;94;Corse
32;30;Gard;76;Occitanie
33;31;Haute-Garonne;76;Occitanie
34;32;Gers;76;Occitanie
35;33;Gironde;75;Nouvelle-Aquitaine

```

Figure 14 Extrait du fichier texte Localisation présent dans l'HDFS

Il est pertinent de noter que pour les tables qui ne nécessitent pas de partitionnement et de clustering seule l'instruction *LOCATION* est nécessaire à leurs peuplements puisque la seconde table, présente dans les autres dimensions, ne trouve pas son utilité ici. Afin de prouver concrètement le peuplement, vous retrouverez l'ensemble des captures d'écran montrant des extraits des dimensions en annexe de ce document.

Faits

Voici la requête de création de la table de faits :

```
CREATE EXTERNAL TABLE IF NOT EXISTS Fait(  
  Id_patient int,  
  Id_professionnel int,  
  Id_diagnostic int,  
  Id_localisation int,  
  Id_temps int,  
  Nombre_admission int,  
  Nombre_hospitalisation int,  
  Nombre_consultation int,  
  Nombre_deces int,  
  Total_satisfaction float)  
COMMENT 'Table de faits'  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\';'  
STORED AS TEXTFILE  
LOCATION '/user/hive/datawarehouse/fait';
```

Figure 16 Création de la table de fait

Voici un exemple de lignes de la table de faits après alimentation de celle-ci par un Job ETL décrit dans la documentation du livrable n°1 :

```
73814;395149;14492;62;3889;0;0;1;56791;  
79500;395151;14493;97;3699;0;0;1;85440;  
96212;395152;14494;94;3987;0;0;1;85440;  
70485;395157;14495;46;3705;0;0;1;34299;76.75268  
97829;395158;14496;94;3283;0;0;1;76829;  
99082;395159;14497;96;3040;0;0;1;76829;  
74255;395160;14498;95;3076;0;0;1;76829;  
94564;395162;14499;35;2478;0;0;1;63374;  
98858;395163;14500;77;2473;0;0;1;75730;  
83542;395164;14501;93;3606;0;0;1;77036;  
58272;395165;14502;93;2943;0;0;1;76829;  
94808;395166;14503;94;3905;0;0;1;85440;  
98878;395169;14504;96;2254;0;0;1;75730;  
87456;395171;14505;87;3030;0;0;1;34993;  
79888;395174;14506;77;2688;0;0;1;76759;  
68998;395175;14507;95;2910;0;0;1;76759;
```

Figure 17 Extrait du fichier texte fait présent dans l'HDFS

(Respectivement : Id_patient, Id_professionnel, Id_diagnostic, Id_localisation, Id_temps, Nombre_admission, Nombre_hospitalisation, Nombre_consultation, Nombre_deces et Total_satisfaction)

Voici ce que les attributs représentent sur une ligne de données :

- Id_patient : Il s'agit d'une clé étrangère vers un patient
- Id_professionnel : Il s'agit d'une clé étrangère vers le professionnel qui a prodigué une consultation au patient (toujours le même si cette consultation a mené à une hospitalisation)
- Id_diagnostic : Il s'agit d'une clé étrangère vers le diagnostic de la consultation du patient (toujours le même si cette consultation a mené à une hospitalisation)
- Id_temps : Il s'agit d'une clé étrangère vers la date à laquelle a eu lieu la consultation ou l'hospitalisation
- Id_localisation : Il s'agit d'une clé étrangère vers le département et la région dans laquelle l'hospitalisation ou la consultation a eu lieu
- Nombre_admission : C'est le nombre d'admission à l'hôpital du patient. Il est présent lors du premier jour de chaque période d'hospitalisation. Il nous permet de compter les hospitalisations de manière unique.
- Nombre_hospitalisation : C'est le nombre d'hospitalisation du patient à la suite de la consultation du professionnel donné, pour le diagnostic donné, à la date donnée et au lieu donné.
- Nombre_consultation : C'est le nombre de consultations du patient prodiguées par le professionnel donné, pour le diagnostic donné, à la date donnée et au lieu donné.
- Nombre_deces : C'est le nombre de décès lié au code région, à la région renseignée dans la localisation et à l'année de la date attachée à la ligne.
- Total_satisfaction : C'est la moyenne des notes de satisfactions liée au code région à la région renseignée dans la localisation et à l'année de la date attachée à la ligne.

III. Requêtes d'analyse

Pour chaque requête d'analyse, nous pouvions choisir entre réaliser des vues sur Hive avec un grand nombre de données, puis réaliser un traitement léger sur Power BI consistant principalement en un tri des données, ou alors nous pouvions effectuer toutes les requêtes d'analyses, plus le filtrage, directement sur Power BI.

Nous avons choisi la première option pour deux raisons : Premièrement, le fait de pouvoir charger un grand nombre de données sur Hive nous permettra de changer facilement les périodes d'analyses depuis Power BI sans avoir à modifier toute la vue. Secondement, on se rend quasi indépendant de notre outil de visualisation, ce qui nous permet une flexibilité sur celui-ci. Si demain par exemple, les analystes décidaient de visualiser les données sur Grafana au lieu de Power BI, ils n'auront que peu de changements à effectuer.

Taux de consultation des patients dans un établissement sur une période

Nous cherchons à connaître le taux de consultation par Etablissement par jour. Pour cela, nous avons besoins de quatre informations :

- La date du jour
- Le nombre de consultations effectuées ce jour-là sur l'ensemble des établissements
- L'établissement concerné
- Le nombre de consultations effectuées ce jour-là sur l'établissement concerné

Voici la requête utilisée pour générer la vue dans Hive :

```
DROP VIEW IF EXISTS TauxConsultationEtablissementVue;
CREATE VIEW TauxConsultationEtablissementVue (Taux, Sum_consultation, Total_consultation,
                                              Libelle_etablissement_professionnel, Annee_temps,
                                              Mois_temps, Jour_temps)
AS SELECT Sum_consultation*100/Total_consultation as Taux, Sum_consultation, Total_consultation,
       Libelle_etablissement_professionnel, Annee_temps, Mois_temps, Jour_temps
FROM (SELECT SUM(Nombre_consultation) AS Sum_consultation, Libelle_etablissement_professionnel, Id_temps
      FROM Fait
      INNER JOIN Professionnel ON Id_professionnel = Id
      WHERE Nombre_consultation != 0
      GROUP BY Libelle_etablissement_professionnel, Id_temps) AS Consultation_etablissement
LEFT JOIN
  (SELECT SUM(Nombre_consultation) AS Total_consultation, Id_temps
   FROM Fait
   WHERE Nombre_consultation != 0
   GROUP BY Id_temps) AS Consultation_total
ON Consultation_etablissement.Id_temps = Consultation_total.Id_temps
LEFT JOIN Temps ON Consultation_etablissement.Id_temps = Temps.Id
WHERE Sum_consultation != 0 AND Total_consultation != 0;
```

Figure 18 Création de la vue du taux de consultation par établissement

Pour l'exploiter dans Power BI, nous pouvons récupérer la colonne taux, calculée par rapport au jour à partir de cette formule :
$$\frac{\text{Somme consultation sur un etablissement}}{\text{Somme consultation sur tous les etablissements}}$$

Dans le cas où l'analyse ne se fait plus par jour mais par mois ou par année, la vue nous fournit les données nécessaires et simplifiées pour pouvoir recalculer le taux rapidement.

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

Nous cherchons à connaître le taux de consultation par diagnostic par jour. Pour cela, nous avons besoins de quatre informations :

- La date du jour
- Le nombre de consultations effectuées ce jour-là sur l'ensemble des établissements
- Le diagnostic concerné
- Le nombre de consultations effectuées ce jour-là pour le diagnostic concerné

Voici la requête utilisée pour générer la vue dans Hive :

```
DROP VIEW IF EXISTS TauxConsultationDiagnosticVue;
CREATE VIEW TauxConsultationDiagnosticVue (Taux, Sum_consultation, Total_consultation,
                                           Libelle_diagnostic, Annee_temps, Mois_temps, Jour_temps)
AS SELECT Sum_consultation*100/Total_consultation AS Taux, Sum_consultation, Total_consultation,
           Libelle_diagnostic, Annee_temps, Mois_temps, Jour_temps
FROM (SELECT SUM(Nombre_consultation) as Sum_consultation, Libelle_diagnostic, Id_temps
      FROM Fait
      LEFT JOIN Diagnostic ON Fait.Id_diagnostic = Diagnostic.Id
      WHERE Nombre_consultation != 0
      GROUP BY Libelle_diagnostic, Id_temps) as Consultation_diagnostic
LEFT JOIN
(SELECT SUM(Nombre_consultation) as Total_consultation, Id_temps
 FROM Fait
 WHERE Nombre_consultation != 0
 GROUP BY Id_temps) as Consultation_total
ON Consultation_diagnostic.Id_temps = Consultation_total.Id_temps
LEFT JOIN Temps ON Consultation_diagnostic.Id_temps = Temps.Id
WHERE Sum_consultation != 0 AND Total_consultation != 0;
```

Figure 19 Création de la vue du taux de consultation par diagnostic

Cette vue nous permettra d'analyser quels sont les diagnostics, accidents ou maladies, qui donnent le plus souvent lieu à une consultation. Cette donnée sera comprise dans le champ Taux, calculée avec : $\frac{\text{Somme consultation sur un code_diagnostic}}{\text{Somme consultation sur tous les diagnostics}}$

Cependant, si la granularité venait à changer, il faudrait alors recalculer ce taux à partir des données brutes aussi fournies.

Taux global d'hospitalisation des patients dans une période donnée

Nous cherchons à connaître le taux de patients qui sont hospitalisés par jour. Pour cela, nous avons besoins de trois informations :

- L'année, l'année et le mois ou la date du jour en fonction de ce que l'on souhaite visualiser
- Le nombre de patients hospitalisés ce jour-là sur l'ensemble des patients enregistrés sur cette journée
- Le nombre de patients enregistrés sur cette journée (pour des hospitalisations comme pour des consultations)

Voici la requête utilisée pour générer la vue dans Hive :

```
DROP VIEW IF EXISTS TauxHospitalisationsGlobalVue;
CREATE VIEW TauxHospitalisationsGlobalVue (Taux, Patient_hospitalisation, Total_patient,
                                           Id_patient, Annee_temps, Mois_temps, Jour_temps)
AS SELECT Patient_hospitalisation*100/Total_patient as Taux, Patient_hospitalisation, Total_patient,
       Id_patient, Annee_temps, Mois_temps, Jour_temps
FROM (SELECT COUNT(Nombre_hospitalisation) AS Patient_hospitalisation, Id_patient, Id_temps
      FROM Fait
      WHERE Nombre_hospitalisation > 0
      GROUP BY Id_patient, Id_temps) AS Patient_hospitalise
LEFT JOIN
  (SELECT COUNT(Id_patient) AS Total_patient, Id_temps
   FROM Fait
   WHERE Nombre_hospitalisation > 0 OR Nombre_consultation > 0
   GROUP BY Id_temps) AS Patient
ON Patient_hospitalise.Id_temps = Patient.Id_temps
LEFT JOIN Temps ON Patient_hospitalise.Id_temps = Temps.Id;
```

Figure 20 Création de la vue du taux d'hospitalisation globale

Cette vue nous permettra d'analyser, en moyenne, combien de consultations donnent lieu à une hospitalisation. Cette donnée sera comprise dans le champ Taux, calculée avec :

$$\frac{\text{Somme des patients hospitalisés}}{\text{Somme de tous les patients de cette période}}$$

Une fois encore, si la période est modifiée, il nous faudra recalculer le taux rapidement à partir des données brutes.

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

Nous cherchons à connaître le taux d'hospitalisations par diagnostic par jour. Pour cela, nous avons besoins de 4 informations :

- La date du jour
- Le nombre de hospitalisations effectuées ce jour-là sur l'ensemble des établissements
- Le diagnostic concerné
- Le nombre de hospitalisations effectuées ce jour-là pour le diagnostic concerné

Voici la requête utilisée pour générer la vue dans Hive :

```
DROP VIEW IF EXISTS TauxHospitalisationsDiagnosticVue;
CREATE VIEW TauxHospitalisationsDiagnosticVue (Taux, Sum_admission, Total_admission,
                                             Libelle_diagnostic, Annee_temps, Mois_temps, Jour_temps)
AS SELECT Sum_admission*100/Total_admission AS Taux, Sum_admission, Total_admission,
       Libelle_diagnostic, Annee_temps, Mois_temps, Jour_temps
FROM (SELECT SUM(Nombre_admission) AS Sum_admission, Libelle_diagnostic, Id_temps
      FROM Fait
      INNER JOIN Diagnostic ON Fait.Id_diagnostic = Diagnostic.Id
      WHERE Nombre_admission != 0
      GROUP BY Libelle_diagnostic, Id_temps) as Admission_diagnostic
INNER JOIN
  (SELECT SUM(Nombre_admission) as Total_admission, Id_temps
   FROM Fait
   INNER JOIN Diagnostic ON Fait.Id_diagnostic = Diagnostic.Id
   GROUP BY Id_temps) as Admission_total
ON Admission_diagnostic.Id_temps = Admission_total.Id_temps
LEFT JOIN Temps ON Admission_diagnostic.Id_temps = Temps.Id
WHERE Sum_admission != 0 AND Total_admission != 0;
```

Figure 21 Création de la vue du taux d'hospitalisation par diagnostic

Cette vue nous permettra d'analyser quels sont les diagnostics, accidents ou maladies, qui donnent le plus souvent lieu à une hospitalisation. Cette donnée sera comprise dans le champ Taux, calculée avec :
$$\frac{\text{Somme des admissions pour un diagnostic}}{\text{Somme de toutes les admissions de la période}}$$

Une fois encore si la période est modifiée, il nous faudra recalculer le taux à partir des données brutes disponibles dans la vue.

Taux d'hospitalisation/consultation par sexe, par âge

Nous cherchons à connaître le taux d'hospitalisations et de consultations par patient en fonction de son sexe et de son âge. Pour cela, nous avons besoins de quatre informations :

- Un âge
- Un Sexe
- Le nombre de consultations liés à des patients de cette tranche d'âge et de ce sexe
- Le nombre d'hospitalisations liés à des patients de cette tranche d'âge et de ce sexe

```

DROP VIEW IF EXISTS ConsultationHospitalisationFemme;
CREATE VIEW ConsultationHospitalisationPatient (Sum_consultation, Sum_hospitalisation, Sexe_patient, Age)
AS SELECT SUM(Nombre_consultation) AS Sum_consultation, SUM(Nombre_hospitalisation) AS Sum_hospitalisation, Sexe_patient, Age
FROM (SELECT Nombre_consultation, Nombre_hospitalisation, Sexe_patient,
IF(Mois_temps < Mois_naissance_patient AND Jour_temps < Jour_naissance_patient,
Annee_temps - Année_naissance_patient - 1, Année_temps - Année_naissance_patient) AS Age
FROM Fait
LEFT JOIN Patient ON Fait.Id_patient = Patient.Id
LEFT JOIN Temps ON Fait.Id_temps = Temps.Id
WHERE Sexe_patient LIKE 'F') AS Tmp
GROUP BY Sexe_patient, Age;

DROP VIEW IF EXISTS ConsultationHospitalisationHomme;
CREATE VIEW ConsultationHospitalisationPatient (Sum_consultation, Sum_hospitalisation, Sexe_patient, Age)
AS SELECT SUM(Nombre_consultation) AS Sum_consultation, SUM(Nombre_hospitalisation) AS Sum_hospitalisation, Sexe_patient, Age
FROM (SELECT Nombre_consultation, Nombre_hospitalisation, Sexe_patient,
IF(Mois_temps < Mois_naissance_patient AND Jour_temps < Jour_naissance_patient,
Annee_temps - Année_naissance_patient - 1, Année_temps - Année_naissance_patient) AS Age
FROM Fait
LEFT JOIN Patient ON Fait.Id_patient = Patient.Id
LEFT JOIN Temps ON Fait.Id_temps = Temps.Id
WHERE Sexe_patient LIKE 'M') AS Tmp
GROUP BY Sexe_patient, Age;

```

Figure 22 Création des vues des taux d'hospitalisation et de consultation par âge et sexe

Pour l'exploiter dans Power BI, il nous suffit juste d'utiliser les chiffres fournis dans la colonne *Sum_consultation* et *Sum_hospitalisation*, de les rassembler par un ensemble d'âge (afin de faire une analyse par tranche d'âge), selon la formule suivante :

$$\frac{\text{Somme des hospitalisations/consultations du sexe et des âges concernés}}{\text{Somme de toutes les hospitalisations/consultations}}$$

Taux de consultation par professionnel

Nous cherchons à connaître le taux de consultation par professionnel de santé par jour. Pour cela, nous avons besoin de six informations :

- La date du jour
- Le nombre de consultations effectuées ce jour-là par l'ensemble des professionnels de santé
- Le Nom d'un professionnel
- Son prénom
- Sa profession
- Le nombre de consultations effectuées ce jour-là par ce professionnel

Voici la requête utilisée pour générer la vue dans Hive :

```

DROP VIEW IF EXISTS TauxConsultationProfessionnelVue;
CREATE VIEW TauxConsultationProfessionnelVue (Taux, Sum_consultation, Total_consultation,
Nom_professionnel, Prenom_professionnel, Profession_professionnel,
Specialite_professionnel, Annee_temps, Mois_temps, Jour_temps)
AS SELECT Sum_consultation*100/Total_consultation AS Taux, Sum_consultation, Total_consultation,
Nom_professionnel, Prenom_professionnel, Profession_professionnel,
Specialite_professionnel, Annee_temps, Mois_temps, Jour_temps
FROM (SELECT SUM(Nombre_consultation) as Sum_consultation, Id_professionnel, Id_temps
FROM Fait
WHERE Nombre_consultation != 0
GROUP BY Id_professionnel, Id_temps) AS Consultation_professionnel
LEFT JOIN
(SELECT SUM(Nombre_consultation) AS Total_consultation, Id_temps
FROM Fait
WHERE Nombre_consultation != 0
GROUP BY Id_temps) AS Consultation_total
ON Consultation_professionnel.Id_temps = Consultation_total.Id_temps
LEFT JOIN Professionnel ON Consultation_professionnel.Id_professionnel = Professionnel.Id
LEFT JOIN Temps ON Consultation_professionnel.Id_temps = Temps.Id;

```

Figure 23 Création de la vue du taux de consultation par professionnel

Une fois cette vue transmise à Power BI, nous aurons la capacité de répondre à la mesure demandée en utilisant le champ Taux qui suit la formule suivante :

$$\frac{\text{Somme des consultations d'un professionnel}}{\text{Somme de toutes les consultations}}$$

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

Nous cherchons à connaître le nombre de décès par région et par année. Pour cela, nous avons besoins de ces quatre informations :

- L'année
- Le Code de la Région
- Le Libellé de la Région
- Le nombre de décès survenus au sein de cette région au cours de l'année identifiée.

Voici la requête utilisée pour générer la vue dans Hive :

```
DROP VIEW IF EXISTS NombreDecesLocalisation;  
CREATE VIEW NombreDecesLocalisation (Nombre_deces, Code_region_localisation, Nom_region_localisation, Annee_temps)  
AS SELECT Nombre_deces, Code_region_localisation, Libelle_region_localisation, Annee_temps  
FROM Fait  
LEFT JOIN Temps ON Fait.Id_temps = Temps.Id  
LEFT JOIN Localisation ON Fait.Id_localisation = Localisation.Id  
GROUP BY Code_region_localisation, Libelle_region_localisation, Annee_temps, Nombre_deces;
```

Figure 24 Création de la vue du nombre de morts par région

La vue nous fournit donc l'ensemble des chiffres de décès, par régions et par années, ce qui nous permet de récupérer rapidement tous les décès de régions liées à une année particulière sur Power BI, ainsi que la possibilité de faire une analyse pluriannuelle.

Taux global de satisfaction par région sur l'année 2020

Nous cherchons à connaître la note de satisfaction par région et par année. Pour cela, nous avons besoins de ces quatre informations :

- L'année
- Le Libellé de la Région
- Le Code de la Région
- La valeur moyenne des notes de satisfactions recensées au sein de cette région au cours de l'année identifiée.

Voici la requête utilisée pour générer la vue dans Hive :

```
DROP VIEW IF EXISTS SatisfactionLocalisation;  
CREATE VIEW SatisfactionLocalisation (Total_satisfaction, Code_region_localisation, Libelle_region_localisation, Annee_temps)  
AS SELECT AVG(Total_satisfaction), Code_region_localisation, Libelle_region_localisation, Annee_temps  
FROM Fait  
LEFT JOIN Localisation ON Fait.Id_localisation = Localisation.Id  
LEFT JOIN Temps ON Fait.Id_temps = Temps.Id  
GROUP BY Code_region_localisation, Libelle_region_localisation, Annee_temps;
```

Figure 25 Création de la vue de satisfaction par région

La vue nous fournit donc l'ensemble des moyennes de notes de satisfactions, par régions et par années, ce qui nous permet de récupérer rapidement toutes les notes de régions liées à une année particulière sur Power BI, ainsi que la possibilité de faire une analyse pluriannuelle.

IV. Evaluation de performance

Afin d'expérimenter les performances de nos choix de partitionnement / bucketing, nous avons testés les sélections faites par nos vues sur Hive afin de comparer les temps de traitements de ces requêtes. Voici le résultat de nos tests, exprimés en secondes :

Requête	Libellé	Avec partition	Sans partition
1	Consultation/établissement/jour	227,293	237,565
2	Consultation/diagnostic/année	148,544	156,247
3	Hospitalisation/patient/année	122,941	129,137
4	Hospitalisation/diagnostic/jour	198,296	210,776
5	Hosp+consul/Homme/âge	105,973	109,904
5	Hosp+consul/Femme/âge	105,507	114,599
6	Consultation/professionnel	192,165	199,529
7	Décès/région/année	156,484	161,727
8	Satisfaction/région/année	55,632	146,113

Figure 26 Résultats des tests de temps d'exécutions des requêtes analytiques

Comme nous pouvons le constater, nous gagnons quelques secondes (excepté pour la dernière requête dont le temps de processus est divisé par trois) de processus pour chacune des requêtes composant nos vues analytiques. Cependant, nos résultats ne permettent pas à eux seuls de confirmer ou de prouver l'efficacité réelle de notre partitionnement, car les sélections, jointures et données qui composent nos vues sont parfois complexes et multiples.

Nous avons donc décidé de faire une requête simple pour montrer l'efficacité de nos choix de partitionnement et bucketing.

Cette requête consiste en la récupération de tous les patients qui sont des femmes et dont l'âge se situe entre vingt et vingt-neuf ans. Voici la requête :

```
SELECT * FROM Patient WHERE Sexe_patient = 'F' AND Tranche_age_patient == 2;
```

La seconde requête est la même pour les hommes :

```
SELECT * FROM Patient WHERE Sexe_patient = 'M' AND Tranche_age_patient == 2;
```

Voici nos résultats :

Tentative	Avec Partition	Sans partition
1	0.174	19.653
2	0.175	26.982

Figure 27 Résultats des tests de temps d'executions d'une requête simplifiée

Ici, la différence est importante : nous diminuons le temps d'exécution de cette requête par cent avec le partitionnement et le bucketing que nous avons choisis.

V. Conclusion

Nous avons à présent réalisé les quatre grandes phases de ce projet que sont l'étude de l'architecture, la modélisation des données, l'implémentation physique des données, leur exploitation ainsi que les mesures de performances associées. Plus précisément, ce livrable nous a permis de détailler :

- La création des tables externes dans Hive pour accueillir les 5 dimensions et la table des faits ainsi que le système de partitionnement/ bucketing utilisé.
- Le peuplement de ces différentes tables
- L'évaluation des performances liées au partitionnement et au bucketing, en comparaison avec des tables sans partitions
- Les requêtes d'analyse, qui vont nous permettre d'afficher les résultats d'analyse de façon claire et visuelle à l'aide de *PowerBI*.

Ainsi, nous serons en mesure de répondre aux différents besoins du groupe CHU et de leur fournir une interprétation fiable des données qu'ils nous ont fournis.

VI. Annexe

chu.diagnostic

Columns

Details

Sample

Analysis

	diagnostic.id	diagnostic.code_diagnostic	diagnostic.libelle_diagnostic
1	1	A066	Abces amibien du cerveau
2	2	A064	Abces amibien du foie
3	3	A065	Abces amibien du poumon
4	4	K610	Abces anal
5	5	K612	Abces anorectal
6	6	L028	Abces cutane, furoncle et anthrax d'autres localisations
7	7	L020	Abces cutane, furoncle et anthrax de la face
8	8	L023	Abces cutane, furoncle et anthrax de la fesse

Table Browser

Figure 28 - Table des diagnostics

/ user / hive / datawarehouse / patient / sexe_patient=F / tranche_age_patient=3 / 000000_0

11;11;20;3;1985
24;24;20;3;1983
33;33;16;9;1987
39;39;18;9;1985
60;60;29;8;1985
74;74;30;10;1986
81;81;2;3;1987
87;87;4;7;1991
00-00-00-11-1000

Figure 29 - Table Patient

	id	identifiant_professionnel	nom_professionnel	prenom_professionnel	identifiant_etablissement_professionnel	libelle_etablissement_professionnel	code_departement_etablisse
1	4673	10000005875	SCHOULER	Bernadette	F970111811	LBM CTRE BIOLOG MEDICALE GRANDE-TERRE	97
2	4717	10000163195	ROI LAUR	Francoise	F780018727	CLINIQUE SAINT GERMAIN	78
3	4720	10000173426	PICHAVANT	Marie Odile		France	48
4	4738	10000250208	DENNERY CHABOT	Claire	F92001183013121999	PHARMACIE NGUYEN DUC	92
5	4739	10000251701	BIENVENU	Claire	F940021645	LBM SELARL BIOMEGA SITE SAINT MAUR RAS	94
6	4740	10000254945	PETHE PAQUEREAU	Laurence	F78000596301062005	PHARMACIE PINAUD-REVEILLERE	78
7	4741	10000262930	BRACONNIER	Claude	F92001761312101972	PHARMACIE MACHELON - BRACONNIER	92
8	4772	10000364405	TISSERAND	Agnes	F95000702112092011	PHARMACIE DURAND-DUPRE	95

Figure 30 - Table Professionnel

chu.temps

Columns Details Sample

	temps.id	temps.jour_temps	temps.mois_temps	temps.annee_temps
1	731	1	1	2012
2	732	2	1	2012
3	733	3	1	2012
4	734	4	1	2012
5	735	5	1	2012
6	736	6	1	2012
7	737	7	1	2012
8	738	8	1	2012

Table Browser

Figure 31 - Table Temps

	id	code_departement_localisation	libelle_departement_localisation	libelle_region_localisation	code_region_localisation
1	4	04	Alpes-de-Haute-Provence	Provence-Alpes-Côte d'Azur	93
2	5	05	Hautes-Alpes	Provence-Alpes-Côte d'Azur	93
3	6	06	Alpes-Maritimes	Provence-Alpes-Côte d'Azur	93
4	13	13	Bouches-du-Rhône	Provence-Alpes-Côte d'Azur	93
5	85	83	Var	Provence-Alpes-Côte d'Azur	93
6	86	84	Vaucluse	Provence-Alpes-Côte d'Azur	93
7	98	970	Outre-Mer	Outre-Mer	970
8	20	20	Corse	Corse	94

Figure 32 - Table Localisation

	id_patient	id_professionnel	id_diagnostic	id_localisation	id_temps	nombre_admission	nombre_hospitalisation	nombre_consultation	nombre_deces
1	5725	4676	1	0	2495	0	0	1	4
2	19907	4683	2	61	3963	0	0	1	56791
3	34905	4692	3	0	3112	0	0	1	4
4	50924	4697	4	0	2696	0	0	1	8
5	1222	4699	5	0	2967	0	0	1	4
6	48461	4702	6	0	2059	0	0	1	31
7	37929	4706	7	77	3208	0	0	1	76829
8	22888	4709	8	0	2468	0	0	1	4
9	10247	4716	9	0	2530	0	0	1	4
10									50

cloudera@quickstart:~

Figure 33 - Table des Faits