

Livable 2 : Modèle physique et optimisation

Projet : BIGDATA

Avril 2024

Mathéo Ricco – Hugo Durupt – Quentin Trappier – Fabien Arrighi
Étudiants au CESI – Cycle Ingénieur Alternance (FISE) Spécialité Informatique

Enseignant référent – M. Rohan FOSSE

Table des matières

| | |
|-----------------------------------|----|
| Tables des figures | 3 |
| Introduction | 4 |
| Contexte | 4 |
| Dimension « Professionnel » | 5 |
| Dimension « Patient » | 8 |
| Dimension « Localisation » | 12 |
| Dimension « Diagnostic » | 15 |
| Dimension « Temps » | 18 |
| Table des « Faits » | 21 |
| Synthèse du document | 23 |
| Conclusion | 24 |

Tables des figures

| | |
|--|----|
| Figure 1 – Visualisation des données de la table « internal_Professionnel » | 5 |
| Figure 2 - Temps d'exécution de la requête externe « Professionnel » sur « Profession » .. | 6 |
| Figure 3 - Temps d'exécution de la requête interne « Professionnel » sur « Profession »... | 6 |
| Figure 4 - Temps d'exécution pour les tables « internal_Professionnel » et « internal_Professionnel_buckets » | 7 |
| Figure 5 - Visualisation des données de la table « internal_Patient » | 8 |
| Figure 6 – Temps d'exécution de la requête externe « Patient » sur « Sexe » | 9 |
| Figure 7 – Temps d'exécution de la requête interne « Patient » sur « Sexe » | 9 |
| Figure 8 – Temps d'exécution de la requête externe « Patient » sur « Age » | 10 |
| Figure 9 - Temps d'exécution de la requête interne « Patient » sur « Age » | 10 |
| Figure 10 - Temps d'exécution pour les tables « internal_Patient » et « internal_Patient_buckets » par « Age » | 10 |
| Figure 11 - Temps d'exécution pour les tables « internal_Patient » et « internal_Patient_buckets » par « Sexe » | 11 |
| Figure 12 - Visualisation des données de la table « internal_Localisation » | 13 |
| Figure 13 - Temps d'exécution de la requête externe « Localisation » sur « Region » | 14 |
| Figure 14 - Temps d'exécution de la requête interne « Localisation » sur « Region » | 14 |
| Figure 15 - Temps d'exécution pour les tables « internal_Localisation » et internal_Localisation_buckets » par « Region » | 14 |
| Figure 16 - Visualisation des données de la table « internal_Diagnostic » | 15 |
| Figure 17 - Temps d'exécution de la requête externe « Diagnostic » sur « Is_Hospitalisation » | 16 |
| Figure 18 - Temps d'exécution de la requête interne « Diagnostic » sur « Is_Hospitalisation » | 17 |
| Figure 19 - Temps d'exécution pour les tables « internal_Diagnostic » et « internal_Diagnostic_buckets » interne par « Is_Hospitalisation » | 17 |
| Figure 20 - Visualisation des données de la table « internal_Temps » | 18 |
| Figure 21 - Temps d'exécution de la requête externe « Temps » sur « Annee » | 19 |
| Figure 22 - Temps d'exécution de la requête interne « Temps » sur « Annee » | 19 |
| Figure 23 - Temps d'exécution pour les tables « internal_Temps » et « internal_Temps_buckets » par « Annee » | 20 |
| Figure 24 - Visualisation des données de la table « internal_Faits » | 21 |

Introduction

Le présent livrable consiste en la mise en place du modèle physique et à l'évaluation de performances par rapport au temps de réponse des requêtes réalisées sur les tables. Nous reviendrons sur la création et le chargement de données dans les tables, nous expliciterons les scripts montrant le peuplement des tables et le partitionnement des buckets. Nous proposerons des graphes montrant les temps de réponses pour évaluer la performance d'accès à l'entrepôt de données. Enfin, nous montrerons des requêtes permettant d'évaluer les performances.

Contexte

La transition numérique s'impose comme une nécessité incontournable pour le secteur de la santé, notamment avec la prise de conscience du potentiel considérable des données générées par les systèmes de gestion des soins et les systèmes FTP. Dans cette optique, le groupe CHU (Cloud Healthcare Unit) se lance dans une transformation digitale majeure. Son objectif principal est de capitaliser sur ces données en mettant en place un entrepôt de données complet capable d'extraire, de stocker, d'explorer et de visualiser les informations selon divers critères. Cette initiative vise à répondre aux besoins variés des utilisateurs, tels que les praticiens et les chefs d'établissement, en matière d'analyse de données pour un suivi efficace des patients à long terme.

Les attentes du groupe CHU sont claires et ambitieuses, ils recherchent une solution intégrée pour l'extraction et le stockage des données, ainsi qu'une consolidation des fichiers distribués dans une source unique. De plus, ils souhaitent une compréhension approfondie des besoins des utilisateurs en matière d'analyse de données, ainsi que des recommandations pertinentes concernant les outils d'intégration, de stockage, de visualisation et d'exploration de données sécurisées pour faciliter la prise de décision.

Cette étude s'engage à relever les défis inhérents au secteur de la santé en proposant une solution répondant aux exigences spécifiques des données médicales, notamment en termes de coût, de sécurité, d'évolutivité et de scalabilité.

Dimension « Professionnel »

Nous allons commencer par la première dimension que nous avons mise en place, la dimension « Professionnel ». Nous avons suivi le MCD que nous vous avons présenté dans le premier livrable.

Dans un premier temps, nous avons créé une table externe issue du fichier TXT que nous avons exporté du job.

```
1. CREATE EXTERNAL TABLE IF NOT EXISTS internal_Professionnel (  
2.   SK_Professionnel INT,  
3.   Nom STRING,  
4.   Prenom STRING,  
5.   Profession STRING  
6. )  
7. ROW FORMAT DELIMITED  
8. FIELDS TERMINATED BY '\';  
9. STORED AS TEXTFILE  
10. LOCATION '/user/hive/data/professionnel'  
11. TBLPROPERTIES ("skip.header.line.count"="1");
```

Voici un exemple de requête pour visualiser les données présentes dans la table.

```
hive> select * from internal_Professionnel LIMIT 20;  
OK  
73      COLLINET      Chantal Assistant de service social  
74      BUKARI Marie Christelle      Assistant de service social  
126     LIGOZAT Solenne Assistant de service social  
196     AUGIER Michele Assistant de service social  
197     MICOULIN      Mireille      Assistant de service social  
198     ISSAUTIER      Laurence      Assistant de service social  
199     FERNANDEZ      Monique Assistant de service social  
200     COSNARD Marie Dominique Assistant de service social  
201     OUESLATI      Radiah Assistant de service social  
202     DAUZET Severine      Assistant de service social  
203     VENTRE Agnes Assistant de service social  
218     CHAREYRE      Bernadette Assistant de service social  
266     GONZALEZ      Marie Laure Assistant de service social  
493     VIANNAY Magali Assistant de service social  
494     MALHAS Nadine Assistant de service social  
495     LAFAGE Christiane Assistant de service social  
496     BLUTEAU Veronique Assistant de service social  
497     FOUGEROLLE      Karine Assistant de service social  
498     MEDORI Sylvie Assistant de service social  
499     DEVILLERS      Emeline Assistant de service social  
Time taken: 0.454 seconds, Fetched: 20 row(s)
```

Figure 1 – Visualisation des données de la table « internal_Professionnel »

Une fois la table externe créée, nous allons nous occuper de créer une table interne en prenant soin de partitionner notre table et d'y ajouter des buckets. Dans le cas de la table « internal_Professionnel », nous avons choisi de la partitionner par « Profession ». Cette partition nous permet d'accéder bien plus rapidement à la liste du personnel de santé d'un même milieu. Pour optimiser la requête ainsi que sauvegarder des données dans la table, nous avons également configuré deux buckets.

```
1. CREATE TABLE internal_Professionnel_buckets (  
2.   SK_Professionnel INT,  
3.   Nom STRING,  
4.   Prenom STRING  
5. )  
6. PARTITIONED BY (Profession STRING)  
7. CLUSTERED BY (SK_Professionnel) into 2 BUCKETS  
8. ROW FORMAT DELIMITED FIELDS TERMINATED BY '\';'  
9. STORED AS TEXTFILE ;
```

Une fois les partitions et les buckets créés, il nous faut remplir la table. Pour ce faire, nous allons nous servir de la table externe et non du fichier TXT pour ajouter une couche supplémentaire de sécurité.

```
1. INSERT OVERWRITE TABLE internal_Professionnel_buckets  
2. PARTITION (Profession)  
3. SELECT SK_Professionnel, Nom, Prenom, Profession  
4. FROM internal_Professionnel ;
```

Pour calculer le temps d'exécution, nous avons utilisé deux requêtes identiques qui seront utiles pour la suite du projet. Voici, ci-dessous, les requêtes ainsi que leur temps d'exécution.

```
1. Select * from internal_Professionnel where Profession="Infirmier" ;  
Time taken: 19.424 seconds, Fetched: 1928 row(s)
```

Figure 2 - Temps d'exécution de la requête externe « Professionnel » sur « Profession »

```
1. Select * from Internal_professionnel_buckets where Profession="Infirmier" ;  
Time taken: 0.11 seconds, Fetched: 1928 row(s)
```

Figure 3 - Temps d'exécution de la requête interne « Professionnel » sur « Profession »

Comme nous pouvons le voir sur ce graphique, le temps d'exécution est plus de 150 fois plus rapide pour la table qui a été partitionnée.

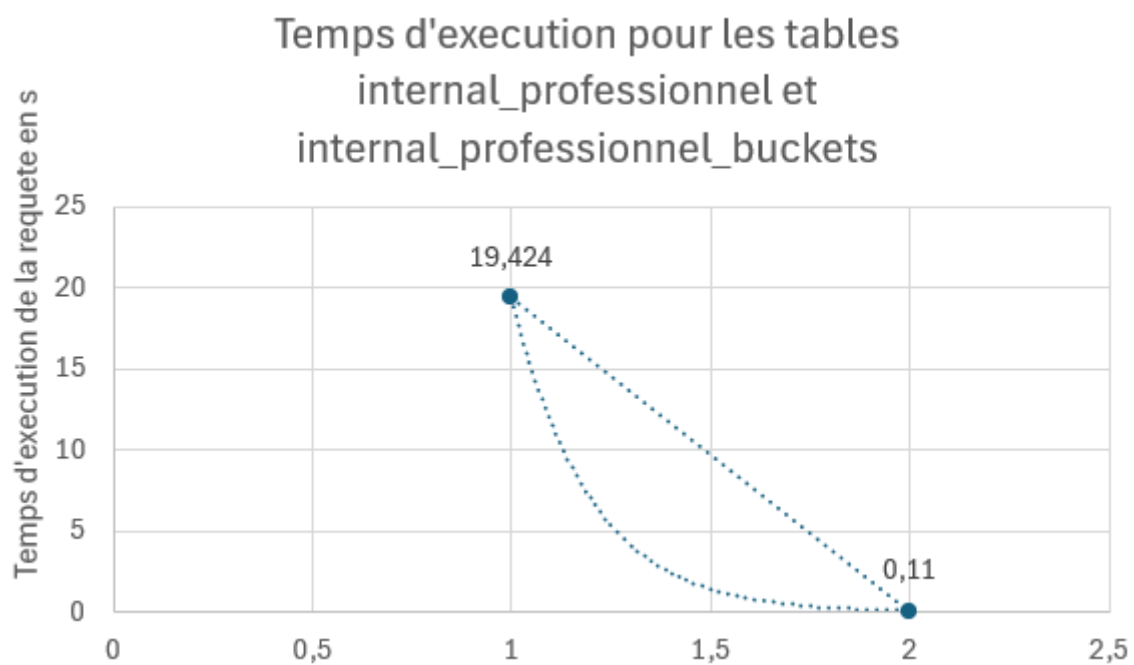


Figure 4 - Temps d'exécution pour les tables « internal_Professionnel » et « internal_Professionnel_buckets »

Dimension « Patient »

Nous allons maintenant voir la dimension « Patient ». Comme pour la dimension « Professionnel », nous avons suivi le MCD que nous vous avons présenté dans le premier livrable.

Dans un premier temps, nous avons, comme précédemment, créé une table externe issue du fichier TXT que nous avons exporté du job.

```
1. CREATE EXTERNAL TABLE internal_Patient (  
2.   SK_PATIENT INT,  
3.   Sexe STRING,  
4.   Age INT  
5. )  
6. ROW FORMAT DELIMITED  
7. FIELDS TERMINATED BY '\;'  
8. STORED AS TEXTFILE  
9. LOCATION '/user/hive/data/internal_Patient'  
10. TBLPROPERTIES (  
11.   "skip.header.line.count"="1"  
12. );
```

Voici un exemple de requête pour visualiser les données présentes dans la table.

```
hive> select * from internal_Patient LIMIT 20;  
OK  
2      female  10  
121    female  10  
463    female  10  
845    female  10  
859    female  10  
937    female  10  
943    female  10  
1176   female  10  
1208   female  10  
1309   female  10  
1419   female  10  
1686   female  10  
1723   female  10  
2087   female  10  
2180   female  10  
2435   female  10  
2543   female  10  
2813   female  10  
3055   female  10  
3086   female  10  
Time taken: 2.472 seconds, Fetched: 20 row(s)
```

Figure 5 - Visualisation des données de la table « internal_Patient »

Une fois la table externe créée, nous allons également nous occuper de créer une table interne en prenant soin de partitionner notre table et d'y ajouter des buckets. Dans le cas de la table « internal_Patient », nous avons choisi de la partitionner par « Sexe » et par « Age ». Cette partition nous permet de catégoriser les individus. Nous sommes partis du postulat que les requêtes se basent très régulièrement sur l'âge et le sexe. Une partition sur ces deux valeurs nous permet donc de réduire énormément le temps de chaque requête. Pour optimiser la requête ainsi que la sauvegarde des données dans la table, nous avons également configuré deux buckets.

```
1. CREATE TABLE internal_Patient_buckets (  
2.   SK_Patient INT  
3. )  
4. PARTITIONED BY (Sexe STRING, Age INT)  
5. CLUSTERED BY (SK_Patient) INTO 2 BUCKETS  
6. ROW FORMAT DELIMITED  
7. FIELDS TERMINATED BY '\';  
8. STORED AS TEXTFILE ;
```

Une fois les partitions et les buckets créés, il nous faut également remplir la table. Pour ce faire, nous allons nous servir de la table externe et non du fichier TXT pour ajouter une couche supplémentaire de sécurité.

```
1. INSERT OVERWRITE TABLE internal_Patient_buckets  
2. PARTITION (Sexe, Age)  
3. SELECT SK_Patient, Sexe, Age  
4. FROM Patient ;
```

Pour calculer le temps d'exécution, nous avons utilisé quatre requêtes identiques qui seront également utiles pour la suite du projet. Voici, ci-dessous, les requêtes ainsi que leurs temps d'exécution.

```
1. Select * from internal_Patient WHERE Sexe="Male" ;
```

Time taken: 22.618 seconds, Fetched: 40781 row(s)

Figure 6 – Temps d'exécution de la requête externe « Patient » sur « Sexe »

```
1. Select * from internal_Patient_buckets WHERE Sexe="Male" ;
```

Time taken: 1.435 seconds, Fetched: 40781 row(s)

Figure 7 – Temps d'exécution de la requête interne « Patient » sur « Sexe »

```
1. Select * from internal_Patient WHERE Age=20 ;
```

Time taken: 19.528 seconds, Fetched: 1012 row(s)

Figure 8 – Temps d'exécution de la requête externe « Patient » sur « Age »

```
1. Select * from internal_Patient_buckets WHERE Age=20 ;
```

Time taken: 0.177 seconds, Fetched: 1012 row(s)

Figure 9 - Temps d'exécution de la requête interne « Patient » sur « Age »

Voici un graphique représentant la différence de temps d'exécution d'une même requête, entre une table non partitionnée et une table partitionnée. Comme nous pouvons le voir, le fait d'avoir choisi de partitionner notre table par « Age », nous a permis de diviser notre temps d'exécution par 110.

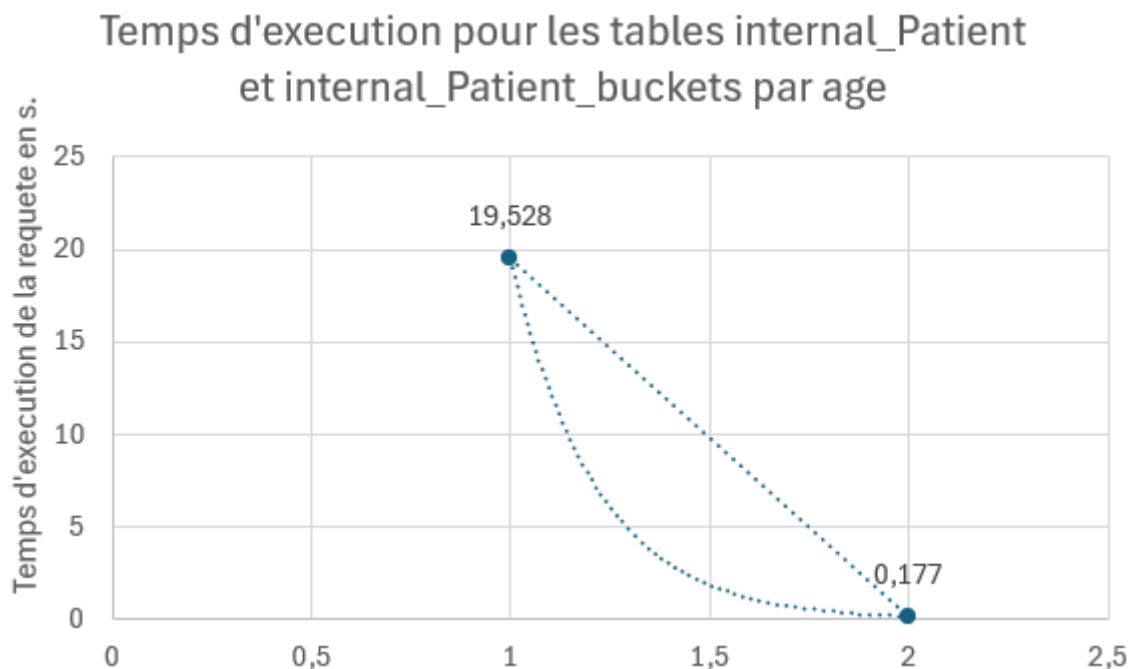


Figure 10 - Temps d'exécution pour les tables « internal_Patient » et « internal_Patient_buckets » par « Age »

Nous pouvons également voir que la partition sur « Sexe » divise le temps d'exécution par 15.

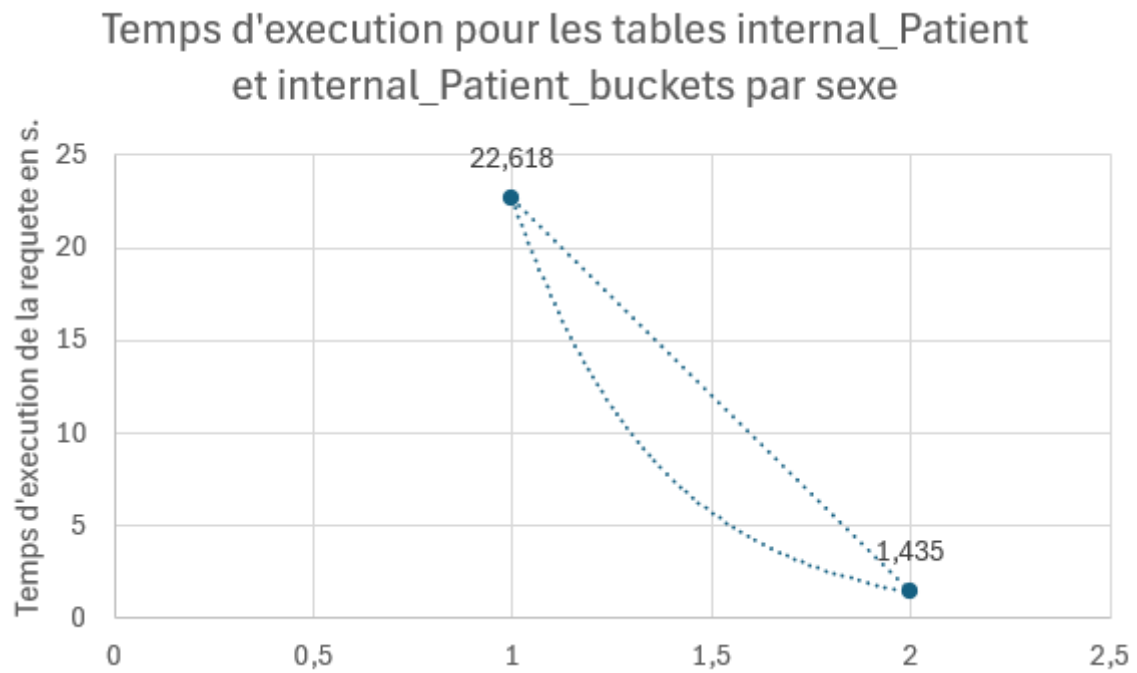


Figure 11 - Temps d'exécution pour les tables « internal_Patient » et « internal_Patient_buckets » par « Sexe »

Dimension « Localisation »

Nous allons maintenant voir la dimension « Localisation ». Comme pour les autres dimensions, nous avons suivi le MCD que nous vous avons présenté dans le premier livrable.

Dans un premier temps, nous avons également créé une table externe issue du fichier TXT que nous avons exporté du job.

```
1. CREATE EXTERNAL TABLE internal_Localisation (  
2.   SK_Localisation INT,  
3.   Etablissement STRING,  
4.   Code_Commune INT,  
5.   Commune STRING,  
6.   Departement STRING,  
7.   Region STRING  
8. )  
9. ROW FORMAT DELIMITED  
10. FIELDS TERMINATED BY '\;  
11. STORED AS TEXTFILE  
12. LOCATION '/user/hive/data/Localisation'  
13. TBLPROPERTIES (  
14.   "skip.header.line.count"="1"  
15. );
```

Voici un exemple de requête pour visualiser les données présentes dans la table :

```
hive> select * from internal_Localisation LIMIT 20;
OK
1030995 EPSAN BRUMATH 67067 BRUMATH BAS-RHIN ALSACE
1030996 EHPAD RESIDENCE DE L'ILLMATT 67028 BENFELD BAS-RHIN ALSACE
1030997 CRLCC PAUL STRAUSS - ICANS 67482 STRASBOURG BAS-RHIN ALSACE
1030998 FAM LE CHATAIGNER ET FAM LE CHARME 67073 CHATENOIS BAS-RHIN ALSACE
1030999 CENTRE HOSPITALIER DE HAGUENAU 67180 HAGUENAU BAS-RHIN ALSACE
1031000 EHPAD MAISON SAINT-JOSEPH 67421 SAALES BAS-RHIN ALSACE
1031001 HOPITAUX UNIVERSITAIRES DE STRASBOURG 67482 STRASBOURG BAS-RHIN ALSACE
1031004 CENTRE DE SOINS INF. HOCHFELDEN 67202 HOCHFELDEN BAS-RHIN ALSACE
1031006 HOPITAL CIVIL / NOUVEL HOPITAL CIVIL 67482 STRASBOURG BAS-RHIN ALSACE
1031007 CENTRE HOSPITALIER DÉPARTEMENTAL 67046 BISCHWILLER BAS-RHIN ALSACE
1031008 CH SAINTE-CATHERINE DE SAVERNE 67437 SAVERNE BAS-RHIN ALSACE
1031009 CENTRE SOINS INF. SELESTAT 67462 SELESTAT BAS-RHIN ALSACE
1031010 HOPITAL DE HAUTEPIERRE 67482 STRASBOURG BAS-RHIN ALSACE
1031012 CLINIQUE RHENA ASSOCIATION 67482 STRASBOURG BAS-RHIN ALSACE
1031013 EFS GRAND EST STRASBOURG 67482 STRASBOURG BAS-RHIN ALSACE
1031014 HOPITAL DU NEUENBERG 67222 INGWILLER BAS-RHIN ALSACE
1031015 C.S.I. CRF DE DRULINGEN 67105 DRULINGEN BAS-RHIN ALSACE
1031016 CLINIQUE SAINTE-ANNE 67482 STRASBOURG BAS-RHIN ALSACE
1031017 MAS OBERKIRCH 67482 STRASBOURG BAS-RHIN ALSACE
1031019 CLINIQUE SAINTE-ODILE 67180 HAGUENAU BAS-RHIN ALSACE
Time taken: 0.62 seconds, Fetched: 20 row(s)
```

Figure 12 - Visualisation des données de la table « internal_Localisation »

Une fois la table externe créée, nous allons également nous occuper de créer une table interne en prenant soin de partitionner notre table et d'y ajouter des buckets. Dans le cas de la table « internal_Localisation », nous avons choisi de la partitionner par « Region ». Cette partition nous permet d'accéder plus rapidement à la liste des différentes localisations d'une même région dans l'objectif d'avoir plus tard le nombre de décès par région. Pour optimiser la requête ainsi que la sauvegarde des données dans la table, nous avons également configuré deux buckets.

```
1. CREATE TABLE internal_Localisation_buckets (
2.   SK_Localisation INT,
3.   Etablissement STRING,
4.   Code_Commune INT,
5.   Commune STRING,
6.   Departement STRING
7. )
8. PARTITIONED BY (Region STRING)
9. CLUSTERED BY (SK_Localisation) INTO 2 BUCKETS
10. ROW FORMAT DELIMITED
11. FIELDS TERMINATED BY '\;'
12. STORED AS TEXTFILE;
```

Une fois les partitions et les buckets créés, il nous faut également remplir la table. Pour ce faire, nous allons nous servir de la table externe et non du fichier TXT pour rajouter une couche supplémentaire de sécurité.

```

1. INSERT OVERWRITE TABLE internal_Localisation_buckets
2. PARTITION (Region)
3. SELECT SK_Localisation, Etablissement, Code_Commune, Commune, Departement,
Region
4. FROM internal_Localisation ;

```

Pour calculer le temps d'exécution, nous avons utilisé deux requêtes identiques qui seront également utiles pour la suite du projet. Voici, ci-dessous, les requêtes ainsi que leur temps d'exécution.

```

1. Select * from internal_Localisation where Region="RHONE-ALPES";
Time taken: 21.745 seconds, Fetched: 26088 row(s)

```

Figure 13 - Temps d'exécution de la requête externe « Localisation » sur « Region »

```

1. Select * from internal_Localisation_buckets where Région="AUVERGNE";
Time taken: 0.119 seconds, Fetched: 26088 row(s)

```

Figure 14 - Temps d'exécution de la requête interne « Localisation » sur « Region »

Comme nous pouvons le voir, le choix de partitionner notre table par « Region » nous a permis de diviser notre temps d'exécution par plus de 180.

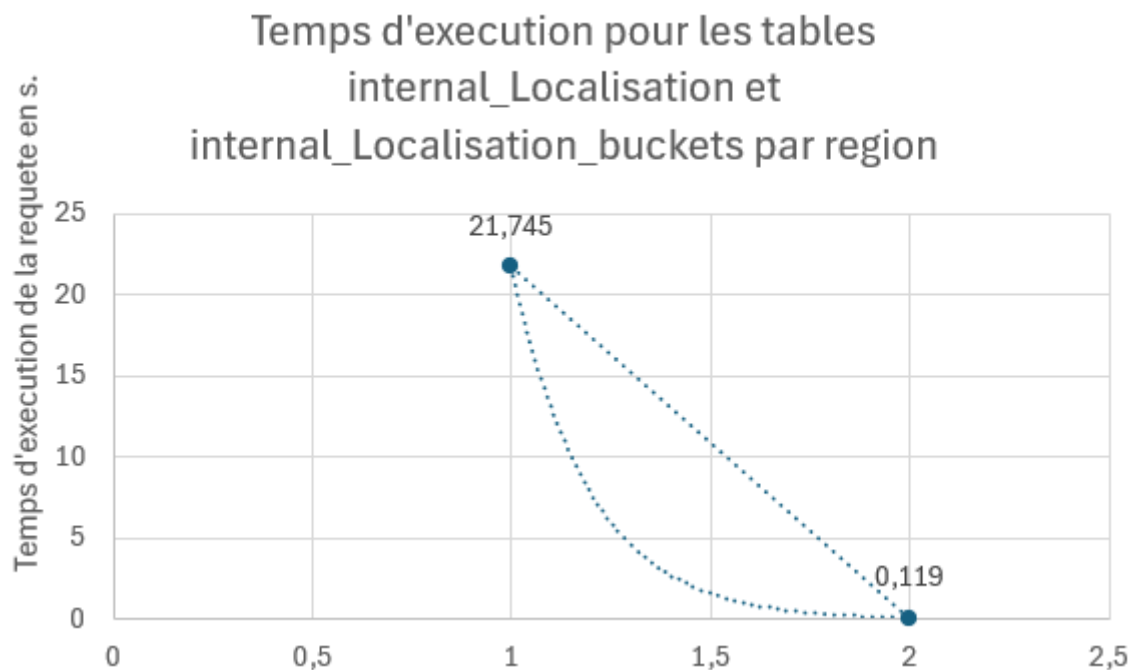


Figure 15 - Temps d'exécution pour les tables « internal_Localisation » et « internal_Localisation_buckets » par « Region »

Dimension « Diagnostic »

Nous allons maintenant voir la dimension « Diagnostic ». Comme pour les autres dimensions, nous avons suivi le MCD que nous vous avons présenté dans le premier livrable.

Dans un premier temps, nous avons également créé une table externe issue du fichier TXT que nous avons exporté du job.

```
1. CREATE EXTERNAL TABLE internal_Diagnostic (  
2.   SK_Diagnostic INT,  
3.   Diagnostic STRING,  
4.   Is_Hospitalisation INT  
5. )  
6. ROW FORMAT DELIMITED  
7. FIELDS TERMINATED BY '\;'  
8. STORED AS TEXTFILE  
9. LOCATION '/user/hive/data/projet_Diagnostic'  
10. TBLPROPERTIES (  
11.   "skip.header.line.count"="1"  
12. );
```

Voici un exemple de requête pour visualiser les données présentes dans la table :

```
hive> select * from internal_Diagnostic LIMIT 20;  
OK  
1      Foetus et nouveau-ne affectes par d'autres medicaments absorbes par la mere      0  
2      Affection inflammatoire pelvienne e Chlamydia de la femme      0  
3      Fractures fermees multiples du pied      0  
4      Autre instabilite articulaire, partie superieure du bras      0  
5      Deplacement d'un autre disque intervertebral precise      0  
6      Infection des voies respiratoires superieures, sans precision      0  
7      Miliaire apocrine      0  
8      Arthropathie post-vaccinale, cheville et pied      0  
9      Scoliose neuromusculaire, localisations vertebrales multiples      0  
10     Corps etranger laisse accidentellement dans l'organisme au cours d'une injection ou vaccination 0  
11     Lancer      0  
12     Hydrocele, sans precision      0  
13     Autres formes de cardiopathie ischemique chronique      0  
14     Syndrome postthrombotique avec ulcere et inflammation      0  
15     Autres anomalies de la continuite osseuse, avant-bras      0  
16     Victime d'une inondation      0  
17     Autres perforations marginales du tympan      0  
18     Dermatose purpurique pigment      0  
19     Affections inflammatoires du scrotum      0  
20     Osteophyte, main      0  
Time taken: 0.136 seconds, Fetched: 20 row(s)
```

Figure 16 - Visualisation des données de la table « internal_Diagnostic »

Une fois la table externe créée, nous allons également nous occuper de créer une table interne en prenant soin de partitionner notre table et d'y ajouter des buckets. Dans le cas de la table «internal_Diagnostic», nous avons choisi de la partitionner par «Is_Hospitalisation». Cette partition nous permet de savoir si un patient est hospitalisé ou non beaucoup, plus rapidement. Pour optimiser la requête ainsi que la sauvegarde des données dans la table, nous avons également configuré deux buckets.

```
1. CREATE TABLE internal_Diagnostic_buckets (  
2.   SK_Diagnostic INT,  
3.   Diagnostic STRING  
4. )  
5. PARTITIONED BY (Is_Hospitalisation INT)  
6. CLUSTERED BY (SK_Diagnostic) INTO 2 BUCKETS  
7. ROW FORMAT DELIMITED  
8. FIELDS TERMINATED BY '\';  
9. STORED AS TEXTFILE;
```

Une fois les partitions et les buckets créés, il nous faut également remplir la table. Pour ce faire, nous allons nous servir de la table externe et non du fichier TXT pour ajouter une couche supplémentaire de sécurité.

```
1. INSERT OVERWRITE TABLE internal_Diagnostic_buckets  
2. PARTITION(Is_Hospitalisation)  
3. SELECT SK_Diagnostic, Diagnostic, Is_Hospitalisation  
4. FROM internal_Diagnostic;
```

Pour calculer le temps d'exécution nous avons utilisé deux requêtes identiques qui seront également utiles pour la suite du projet. Voici, ci-dessous, les requêtes ainsi que leur temps d'exécution :

```
1. Select * from internal_Diagnostic WHERE Is_Hospitalisation=1;  
Time taken: 20.087 seconds, Fetched: 2479 row(s)
```

Figure 17 - Temps d'exécution de la requête externe « Diagnostic » sur « Is_Hospitalisation »


```
1. Select * from internal_Diagnostic_buckets WHERE Is_Hospitalisation=1;
```

Time taken: 0.105 seconds, Fetched: 2479 row(s)

Figure 18 - Temps d'exécution de la requête interne « Diagnostic » sur « Is_Hospitalisation »

Comme nous pouvons le voir, le choix de partitionner notre table par « Is_Hospitalisation » nous a permis de diviser notre temps d'exécution par plus de 190.

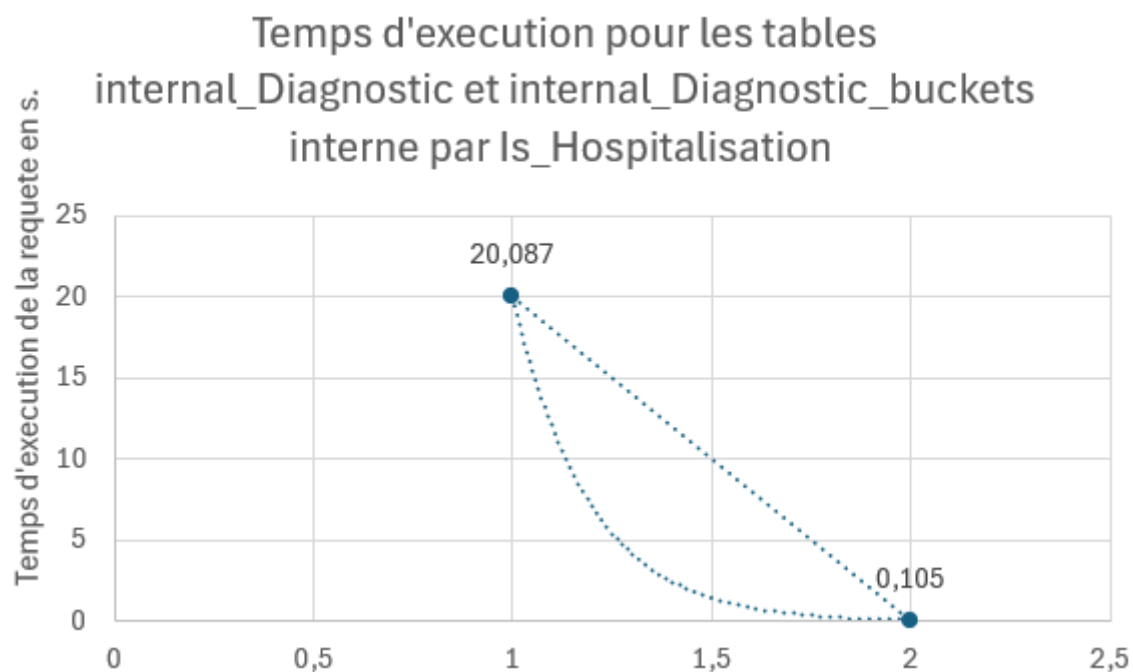


Figure 19 - Temps d'exécution pour les tables « internal_Diagnostic » et « internal_Diagnostic_buckets » interne par « Is_Hospitalisation »

Dimension « Temps »

Nous allons maintenant voir la dimension « Temps ». Comme pour les autres dimensions, nous avons suivi le MCD que nous vous avons présenté dans le premier livrable.

Dans un premier temps, nous avons également créé une table externe issue du fichier TXT que nous avons exporté du job.

```
1. CREATE EXTERNAL TABLE internal_Temps (  
2.   SK_Date INT,  
3.   Date DATE,  
4.   Annee INT  
5. )  
6. ROW FORMAT DELIMITED  
7. FIELDS TERMINATED BY '\;'  
8. STORED AS TEXTFILE  
9. LOCATION '/user/hive/data/Temps1'  
10. TBLPROPERTIES (  
11.   "skip.header.line.count"="1"  
12. );
```

Voici un exemple de requête pour visualiser les données présentes dans la table.

```
hive> select * from internal_Temps LIMIT 20;  
OK  
1      2017-09-27      NULL  
2      2020-08-16      NULL  
3      2016-01-16      NULL  
4      2018-03-24      NULL  
5      2020-04-16      NULL  
6      2021-04-14      NULL  
7      2018-08-05      NULL  
8      2015-12-31      NULL  
9      2017-07-27      NULL  
10     2015-06-25      NULL  
11     2015-06-26      NULL  
12     2020-05-31      NULL  
13     2018-10-12      NULL  
14     2016-04-02      NULL  
15     2021-05-03      NULL  
16     2020-02-14      NULL  
17     2018-09-17      NULL  
18     2019-08-31      NULL  
19     2020-03-11      NULL  
20     2020-04-28      NULL  
Time taken: 0.101 seconds, Fetched: 20 row(s)
```

Figure 20 - Visualisation des données de la table « internal_Temps »

Une fois la table externe créée, nous allons également nous occuper de créer une table interne en prenant soin de partitionner notre table et d'y ajouter des buckets. Dans le cas de la table « internal_Temps », nous avons choisi de la partitionner par « Année ». Cette partition nous permet de trier le plus vite possible les données par année. Pour optimiser la requêtes ainsi que la sauvegarde des données dans la table, nous avons également configuré deux buckets.

```
1. CREATE TABLE internal_Temps_buckets (  
2.   SK_Date INT,  
3.   Date DATE  
4. )  
5. PARTITIONED BY (Annee INT)  
6. CLUSTERED BY (SK_Date) into 2 BUCKETS  
7. ROW FORMAT DELIMITED  
8. FIELDS TERMINATED BY '\;'  
9. STORED AS TEXTFILE;
```

Une fois les partitions et les buckets créés, il nous faut également remplir la table. Pour ce faire, nous allons nous servir de la table externe et non du fichier TXT pour rajouter une couche supplémentaire de sécurité.

```
1. INSERT OVERWRITE TABLE internal_Temps_buckets  
2. PARTITION (Annee)  
3. SELECT SK_Date, Date, Année  
4. FROM internal_Temps;
```

Pour calculer le temps d'exécution, nous avons utilisé deux requêtes identiques qui seront également utiles pour la suite du projet. Voici, ci-dessous, les requêtes ainsi que leurs temps d'exécution.

```
1. Select * from internal_Temps where Année=2019;
```

Time taken: 21.107 seconds, Fetched: 350 row(s)

Figure 21 - Temps d'exécution de la requête externe « Temps » sur « Année »

```
1. Select * from internal_Temps_buckets where Année=2019;
```

Time taken: 0.117 seconds, Fetched: 350 row(s)

Figure 22 - Temps d'exécution de la requête interne « Temps » sur « Année »

Comme nous pouvons le voir, le choix de partitionner notre table par « Année » nous a permis de diviser notre temps d'exécution par plus de 180.

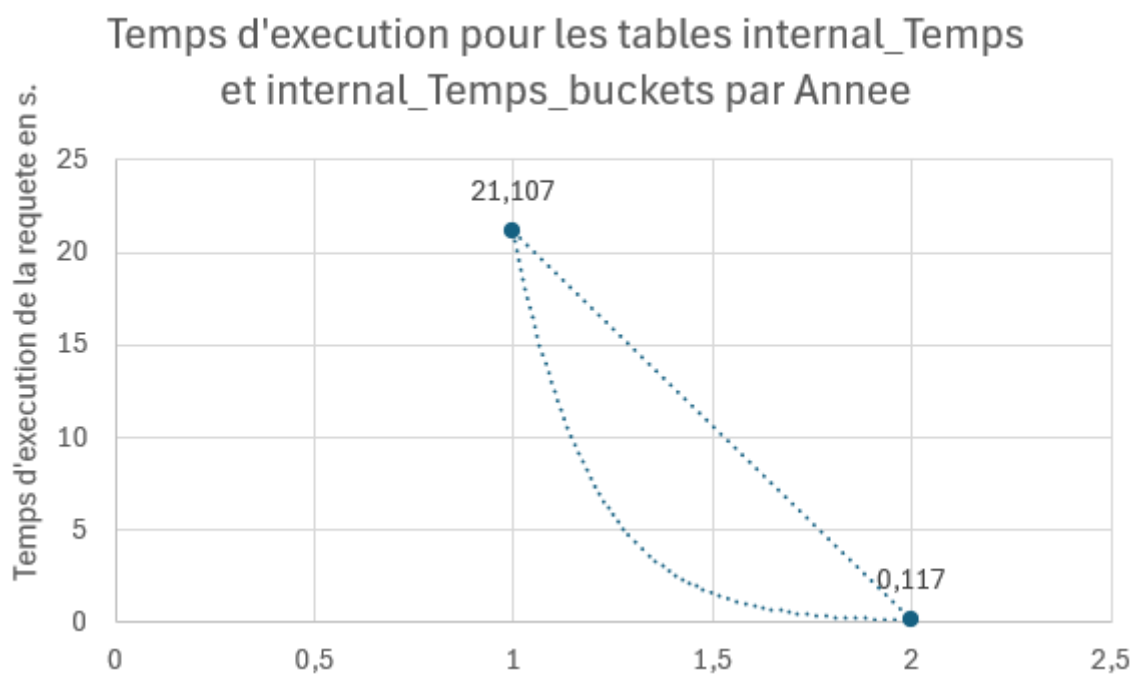


Figure 23 - Temps d'exécution pour les tables « internal_Temps » et « internal_Temps_buckets » par « Année »

Table des « Faits »

Nous allons finir par la création de la table des « Faits ». Nous allons également suivre le modèle défini dans notre MCD que nous vous avons présenté lors du premier livrable.

Dans un premier temps, nous avons également créé une table externe issue du fichier TXT que nous avons exporté du job.

```
1. CREATE EXTERNAL TABLE internal_Faits (  
2.   SK_Diagnostic INT,  
3.   SK_Patient INT,  
4.   SK_Date INT,  
5.   SK_Professionnel INT,  
6.   SK_Localisation INT,  
7.   Nb_Morts INT,  
8.   Satisfaction INT  
9. )  
10. ROW FORMAT DELIMITED  
11. FIELDS TERMINATED BY ';'   
12. STORED AS TEXTFILE  
13. LOCATION '/user/hive/data/projet_Faits'  
14. TBLPROPERTIES (  
15.   "skip.header.line.count"="1"  
16. );
```

Voici un exemple de requête pour visualiser les données présentes dans la table.

```
hive> select * from internal_Faits WHERE SK_Patient>=0 LIMIT 20;  
OK  
1      273      1213      915121      2128      NULL      NULL  
2      1072      1213      788053      2128      NULL      NULL  
3      1286      1213      497390      416679      NULL      NULL  
4      3724      1213      844514      8630      NULL      NULL  
5      3842      1213      963327      205      NULL      NULL  
6      4093      1213      914509      416679      NULL      NULL  
7      4165      1213      907933      240110      NULL      NULL  
8      4478      1213      1020522      416679      NULL      NULL  
9      4710      1213      547095      416679      NULL      NULL  
10     4790      1213      272821      1592      NULL      NULL
```

Figure 24 - Visualisation des données de la table « internal_Faits »

Une fois la table externe créée, nous allons également nous occuper de créer une table interne en prenant soin de partitionner notre table et d'y ajouter des buckets. Dans le cas de la table « Faits », nous avons choisi de la partitionner par « Nb_Morts » et par « Satisfaction ». Cette partition nous permet de trier le plus vite possible les données par nombre de morts et satisfaction. Pour optimiser la requête, ainsi que la sauvegarde des données dans la table, nous avons également configuré deux buckets.

```
1. CREATE TABLE internal_Faits_buckets (  
2.   SK_Diagnostic INT,  
3.   SK_Patient INT,  
4.   SK_Date INT,  
5.   SK_Professionnel INT,  
6.   SK_Localisation INT  
7. )  
8. PARTITIONED BY (Nb_Morts INT, Satisfaction INT)  
9. ROW FORMAT DELIMITED  
10. FIELDS TERMINATED BY ';' ;  
11. STORED AS TEXTFILE ;
```

Une fois les partitions et les buckets créés, il nous faut également remplir la table. Pour ce faire, nous allons nous servir de la table externe et non du fichier TXT pour ajouter une couche supplémentaire de sécurité.

```
1. INSERT OVERWRITE TABLE internal_Faits_buckets  
2. PARTITION(Nb_Morts, Satisfaction)  
3. SELECT  
4.   SK_Diagnostic,  
5.   SK_Patient,  
6.   SK_Date,  
7.   SK_Professionnel,  
8.   SK_Localisation,  
9.   Nb_Morts,  
10.  Satisfaction  
11. FROM internal_Faits;
```

Synthèse du document

Tout le long de ce document, nous avons pu observer l'utilité des partitions. Le fait de partitionner chacune des dimensions nous a permis de gagner énormément de temps sur l'exécution des commandes spécifique. De plus les buckets nous on permit de grandement optimiser le stockage et la fluidité des données.

Conclusion

Au fil de ce document, nous avons exploré la création des diverses bases de données dans le Datawarehouse. Nous les avons partitionnées afin d'optimiser les performances lors des requêtes qui y sont effectuées. Ensuite, nous les avons peuplées avec les données stockées sur l'HDFS, importées précédemment depuis Talend. Nous sommes désormais prêts à entamer l'analyse de ces différentes tables. Pour ce faire, nous utiliserons l'outil Power BI qui nous permettra de présenter efficacement nos résultats relatifs aux différents besoins du projet.