23/10/2025

Livrable 2

Modèle physique et optimisation

Alban CALVO – Evan JOASSON – Mathéo PINGET
FISA INFO 24 - 27

Table des matières

1. Introduction

Le potentiel énorme associé aux données médicales a conduit le secteur de la santé à une transformation importante et rapide. Ainsi, les exigences de mise en place d'améliorations sont de plus en plus significatives. Pour progresser dans la bonne voie, les praticiens (médecins, personnel infirmier) et les administrateurs d'établissements doivent pouvoir accéder directement aux informations exploitables dans les données médicales, afin d'améliorer leurs performances et la qualité des soins de manière mesurable. La demande en matière d'informations exploitables découlant des données médicales intégrées se fait pressante. Les données relatives à l'affluence des patients, aux dossiers médicaux, au suivi des services, aux durées des cycles et sur la rentabilité des établissements recèlent d'importantes informations encore inexploitées, qui attendent d'être découvertes.

Pour toutes ces raisons, le secteur doit investir dans le développement des systèmes informatiques évolutifs, qui comprennent un ensemble d'outils et de mécanismes pour charger, extraire et traiter les données médicales. Ces systèmes parient sur la puissance de traitement parallèle et des ressources distribuées pour effectuer des transformations et des analyses complexes, pour optimiser la prise de décisions. Ainsi, pour rendre cette analyse possible, les entreprises ont besoin de nouvelles données agrégées, consolidées, historiques et synthétisées selon plusieurs axes. Le besoin d'une nouvelle architecture, qui stocke et traite ce type de données a émergé du domaine de la BI et a donné naissance aux infrastructures d'entreposage de données (datawarehouse et datamart). Ces dernières sont des structures qui intègrent des données pour l'analyse et la présentation d'informations pertinentes.

Par ailleurs, le développement d'un système décisionnel évolutif fait face à une série de défis techniques dont le paradigme des données massives (Big Data). Tout d'abord, en raison de la variété et du volume des sources de données disparates, il est difficile de recueillir et d'intégrer des données à partir d'emplacements distribués. Deuxièmement, les systèmes de données volumineux doivent stocker et gérer l'ensemble des données massives et hétérogènes recueillies et assurer une certaine performance en termes d'accès rapide, d'évolutivité et de protection de la vie privée.

2. Objectifs du livrable 2

Ce livrable s’inscrit dans la deuxième phase du projet et a pour objectif de **passer du modèle logique au modèle physique** des données.

Il vise à :

1. Concevoir et implémenter le **modèle physique** des tables de l’entrepôt de données ;

2. Développer les **scripts de création et de chargement** des données dans les tables ;
3. Vérifier l'**intégrité et l'accessibilité** des données stockées ;
4. Mettre en œuvre des **mécanismes d'optimisation** tels que le **partitionnement** et le **bucketing** ;
5. Évaluer les **performances** à travers les temps de réponse des requêtes, avant et après optimisation.

2.1 Technologies utilisées

- **Docker** : pour le déploiement et la gestion des environnements isolés.
- **PostgreSQL** : base relationnelle principale, source de données structurées.
- **MinIO** : stockage objet compatible S3, servant de dépôt pour les fichiers CSV et les résultats analytiques.
- **Jupyter Notebook** : environnement interactif utilisé pour exécuter les scripts Python et SQL, mesurer les temps de réponse et visualiser les résultats.
- **Python** : pour l'automatisation des chargements, la manipulation des données et l'analyse des performances.

3. Modèle physique de données

3.1 Objectifs du modèle physique

Le **modèle physique** traduit le modèle conceptuel et logique en une implémentation concrète dans le système de gestion de base de données choisi, ici **PostgreSQL**.

Il définit :

- Les **tables physiques** de l'entrepôt de données,
- La **nature des relations** entre ces tables (clés primaires et étrangères),
- Les **types de données** utilisés,
- Ainsi que les **mécanismes d'optimisation** (index, partitionnement, stockage).

L'objectif est d'obtenir une structure **performante**, **cohérente** et **optimisée pour la lecture**, adaptée aux besoins d'analyse des utilisateurs finaux (praticiens, chefs d'établissement).

3.2 Architecture en étoile

Le modèle retenu suit une **architecture en étoile**, classique dans les entrepôts de données décisionnels. Ce choix permet de faciliter les **requêtes analytiques** (agrégations, filtres, regroupements) tout en réduisant le nombre de jointures nécessaires.

Tables de faits :

- Consultation
- Décès
- Hospitalisation
- Activité temporelle

Métriques :

- Consultation
- Décès démographie
- Hospitalisation établissement

Tables de dimensions :

- Etablissement
- Patient
- Temps
- Professionnel de santé

4. Zone bronze

La couche **Bronze** constitue le premier niveau du pipeline de traitement des données. Son rôle principal est de **centraliser, nettoyer, normaliser et anonymiser** les données brutes issues de plusieurs sources hétérogènes avant leur exploitation analytique.

Objectif :

Garantir une ingestion fiable et traçable de données provenant de :

- Fichiers **CSV** (open data, établissements, satisfaction, qualité, etc.)
- Bases **PostgreSQL** (patients, consultations, diagnostics, etc.)

L'objectif est de disposer d'une version unifiée et techniquement exploitable des données tout en respectant les contraintes de confidentialité (RGPD).

Étapes de traitement

1. **Initialisation de la session Spark**
    - Configuration optimisée pour ressources limitées (2 Go RAM, 2 cœurs).
    - Chargement dynamique des dépendances JDBC et S3.
    - Tests automatiques de connexion à MinIO et PostgreSQL.
2. **Lecture des données**
    - Extraction depuis PostgreSQL (via JDBC) ou CSV selon la configuration.
    - Filtrage spécifique (ex. : décès uniquement pour l'année 2019).
3. **Nettoyage et standardisation**
    - Uniformisation des noms de colonnes.
    - Normalisation des formats de date (dd/MM/yyyy, yyyy-MM-dd, etc.).
    - Harmonisation des valeurs (sexe, code postal, téléphone, email).
4. **Anonymisation ciblée**
    - Hachage SHA-256 pour les informations personnelles identifiables (nom, prénom, adresse, etc.).
    - Conservation de certaines caractéristiques utiles à l'analyse (initiale du prénom, code département, etc.).
5. **Génération des clés de substitution**
    - Création d'identifiants uniques (_sk) pour chaque dimension : patient, professionnel, diagnostic, établissement, région, etc.
    - Ces clés assurent la cohérence des jointures dans les couches supérieures (Silver et Gold).
6. **Ajout des métadonnées techniques**
    - Colonnes de suivi : _hash_record, _ingestion_date, _batch_id, _source_system, etc.
    - Gestion des versions et états (_is_current, *is_deleted*).
7. **Écriture dans MinIO**
    - Données écrites au format **Parquet** (compression Snappy).
    - Structure : s3a://bronze/<nom_table>/.
    - Regroupement des fichiers par partition logique (2 fichiers maximum/table).

Résultat

La couche Bronze produit un **ensemble de tables Parquet structurées et anonymisées**, prêtes pour les enrichissements et agrégations de la couche Silver.

5. Zone silver

La **zone Silver** sert à transformer et enrichir les données brutes de la couche Bronze pour les rendre **conformes, harmonisées et prêtes pour l'analyse avancée**. Elle constitue la base pour la création des tables Gold et des métriques business.

1.1 Dimensions

Dim_patient

Champ	Type	Description
patient_sk	STRING	Clé substituée unique du patient (technique)
patient_nk	STRING	Clé naturelle provenant de la Bronze table id_patient
nom	STRING	Nom du patient (anonymisé)
prenom	STRING	Prénom du patient (anonymisé)
sexe	STRING	Sexe du patient
date_naissance	DATE	Date de naissance
current_year	INTEGER	Année courante utilisée pour calculs
age	INTEGER	Âge calculé à partir de la date de naissance
tranche_age	STRING	Catégorie standardisée d'âge (0-17, 18-35, etc.)
ville_normalisee	STRING	Ville formatée en majuscules et sans espaces
departement_code	STRING	Code du département dérivé du code postal
silver_created_at	TIMESTAMP	Date et heure de création dans Silver
source_layer	STRING	Indique la couche d'origine ("silver_layer")
is_active	BOOLEAN	Indique si le patient est actif

Dim_etablissement

Champ	Type	Description
etablissement_sk	STRING	Clé substituée unique
etablissement_nk	STRING	Clé naturelle de l'établissement (identifiant_organisation)
nom_etablissement	STRING	Nom du site ou de l'établissement
type_etablissement	STRING	Catégorie standardisée (CHU, Hôpital, Clinique, etc.)
commune_normalisee	STRING	Commune en majuscules et sans espaces
departement_normalise	STRING	Département dérivé du code postal
region_normalisee	STRING	Région approximative basée sur le code postal
silver_created_at	TIMESTAMP	Date de création dans Silver
source_layer	STRING	Couche d'origine ("silver_layer")

Dim_temps

Champ	Type	Description
date_complete	DATE	Date complète (séquence journalière)
annee	INTEGER	Année
mois	INTEGER	Mois
trimestre	INTEGER	Trimestre
jour	INTEGER	Jour du mois
jour_semaine	STRING	Jour en format texte (lundi, mardi…)
type_jour	STRING	Semaine ou Weekend

1.2 Faits

Fact_consultation

Champ	Type	Description

patient_sk	STRING	Référence à dim_patient
consultation_nk	STRING	Identifiant unique de la consultation
date_consultation	DATE	Date de la consultation
annee_consultation	INTEGER	Année de la consultation
mois_consultation	INTEGER	Mois de la consultation
nb_consultations	INTEGER	Nombre de consultations (1 par ligne)
diagnostic_code	STRING	Code diagnostic associé
silver_created_at	TIMESTAMP	Date de création dans Silver
source_system	STRING	Table Bronze source

Fact_hospitalisation

Champ	Type	Description
patient_sk	STRING	Référence à dim_patient
etablissement_sk	STRING	Référence à dim_etablissement
hospitalisation_nk	STRING	Identifiant de l'hospitalisation
date_entree	DATE	Date d'entrée à l'hôpital
date_sortie	DATE	Date de sortie (approximée si non disponible)
duree_sejour	INTEGER	Durée en jours
nb_hospitalisations	INTEGER	Nombre (1 par ligne)
diagnostic_principal	STRING	Code diagnostic principal
silver_created_at	TIMESTAMP	Date de création
source_system	STRING	Table Bronze source

Fact_deces

Champ	Type	Description
patient_sk	STRING	Référence patient (si jointure possible)
etablissement_sk	STRING	Référence établissement (si disponible)

deces_nk	STRING	Identifiant unique du décès
date_deces	DATE	Date de décès
annee_deces	INTEGER	Année du décès
age_deces	INTEGER	Âge au moment du décès
nb_deces	INTEGER	Nombre de décès (1 par ligne)
sexe	STRING	Sexe du défunt
silver_created_at	TIMESTAMP	Date de création dans Silver

1.3 Métriques

Table	Objectif	Principaux champs
metrique_consultation	Volume et fréquentation des consultations	annee_consultation, mois_consultation, nb_consultations_total, nb_patients_uniques, taux_frequentation_moyenne
metrique_hospitalisation_etablissement	Analyse des hospitalisations par type et région	type_etablissement, region_normalisee, nb_hospitalisations, duree_sejour_moyenne, nb_patients_uniques
metrique_deces_demographie	Étude des décès par sexe et tranche d'âge	annee_deces, sexe, tranche_age, nb_deces, age_moyen_deces
metrique_activite_temporelle	Suivi de l'activité mensuelle	annee_consultation, mois_consultation, volume_consultations, ratio_consultations_patient

6. Gold

La **zone Gold** constitue la dernière couche du Data Lake. Elle regroupe les **données prêtes à l'analyse** et à la **prise de décision**. Contrairement aux zones précédentes (Bronze et Silver), où les données sont encore brutes ou simplement nettoyées, la zone Gold contient des informations **enrichies, agrégées et structurées selon des besoins métier précis**.

L'objectif principal est de fournir des datasets **fiables, performants et directement exploitables** par les outils de **Business Intelligence (BI)**, de **reporting**, ou encore de **machine learning**.

La zone gold contient 8 tables KPI :

- kpi_taux_consultation_periode : Activité par période
- kpi_taux_consultation_etablissement : Activité par établissement
- kpi_consultation_par_diagnostic : Activité par pathologie
- kpi_taux_consultation_periode : Activité par période
- kpi_taux_consultation_etablissement : Activité par établissement
- kpi_consultation_par_diagnostic : Activité par pathologie
- kpi_deces_par_region_2019 : Décès par région
- kpi_deces_par_region_2019 : Décès par région

La compression des données assure qu'il n'y ai pas de doublons et de données invalides. On obtient un taux de compression de 99%.

6.1 Indicateurs métiers

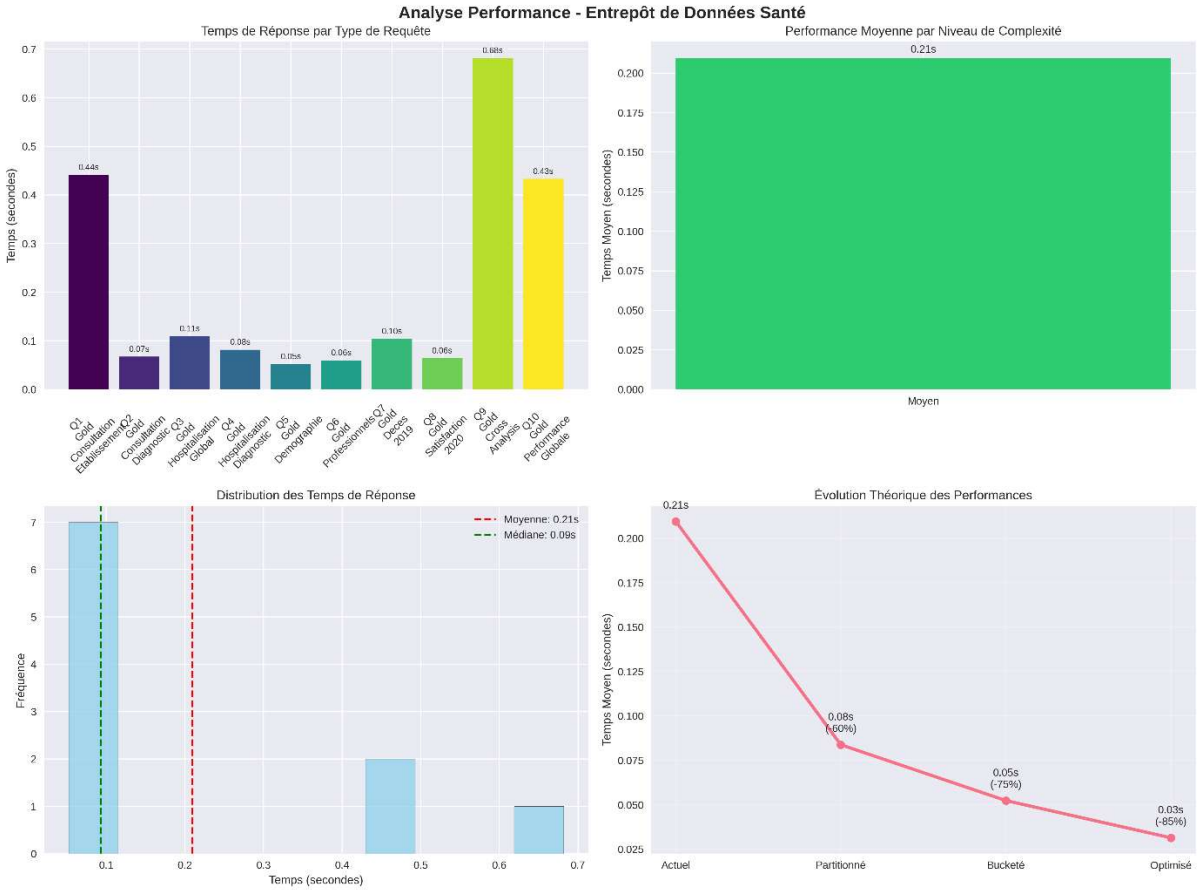

Figure 1 : Analyse de performances

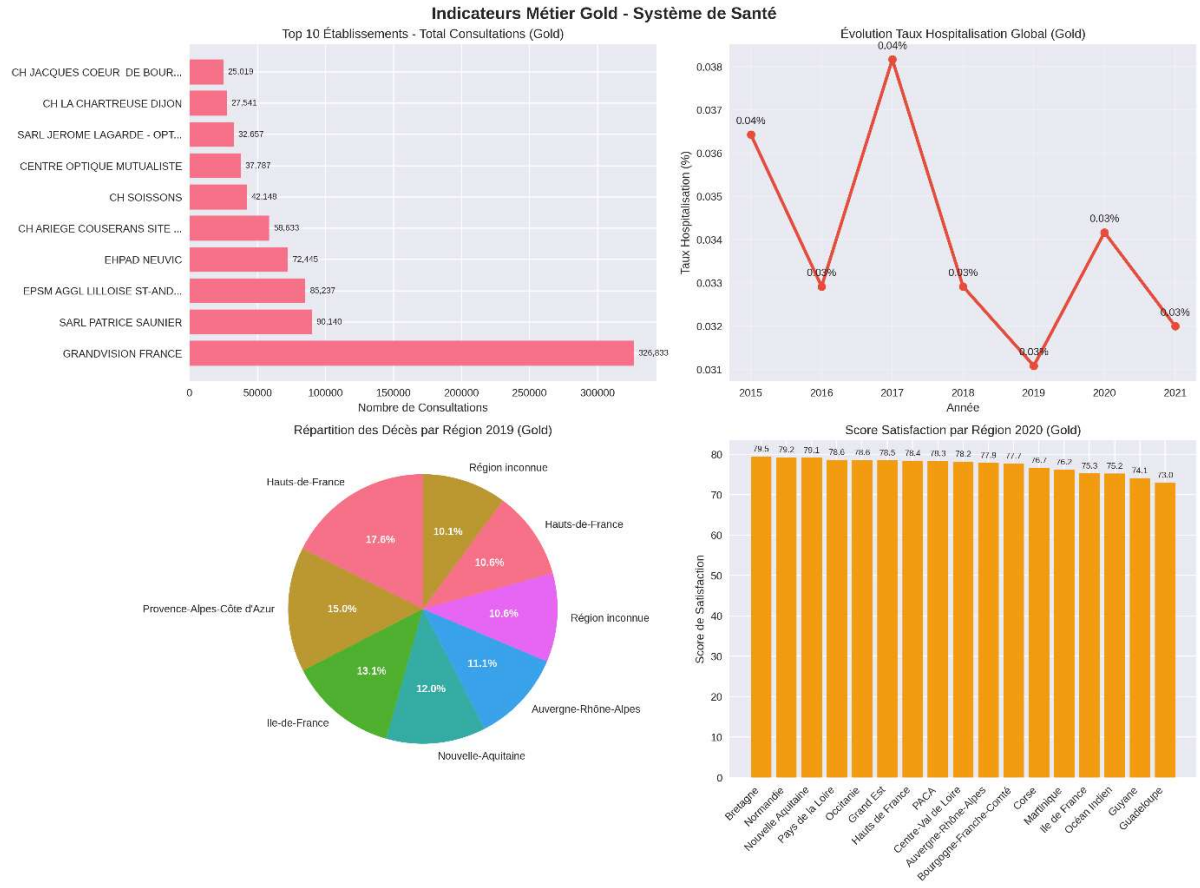

Figure 2 : Indicateurs métier

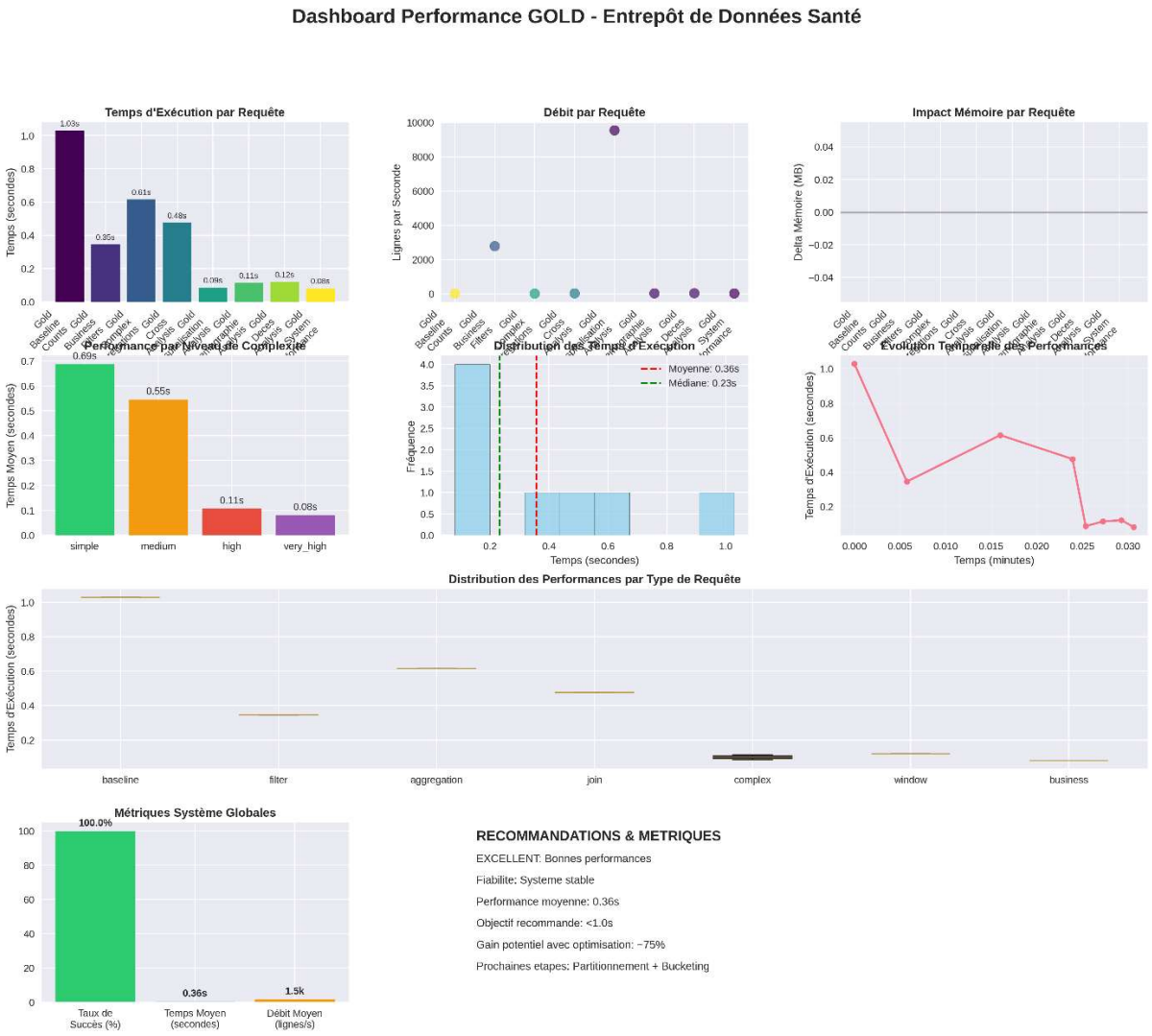

Figure 3 : Performance de l'entrepôt

7. Performances

Volumes de données par zone

Zone	Nombre de tables	Lignes totales	Colonnes totales	Stockage estimé
Bronze	28	7,616,603 lignes	627 colonnes	~726 MB
Silver	10	2,169,531 lignes	77 colonnes	~207 MB
Gold	12	1,563 lignes	55 colonnes	~0.03 MB

7.1 Indicateurs et performances

Zone bronze

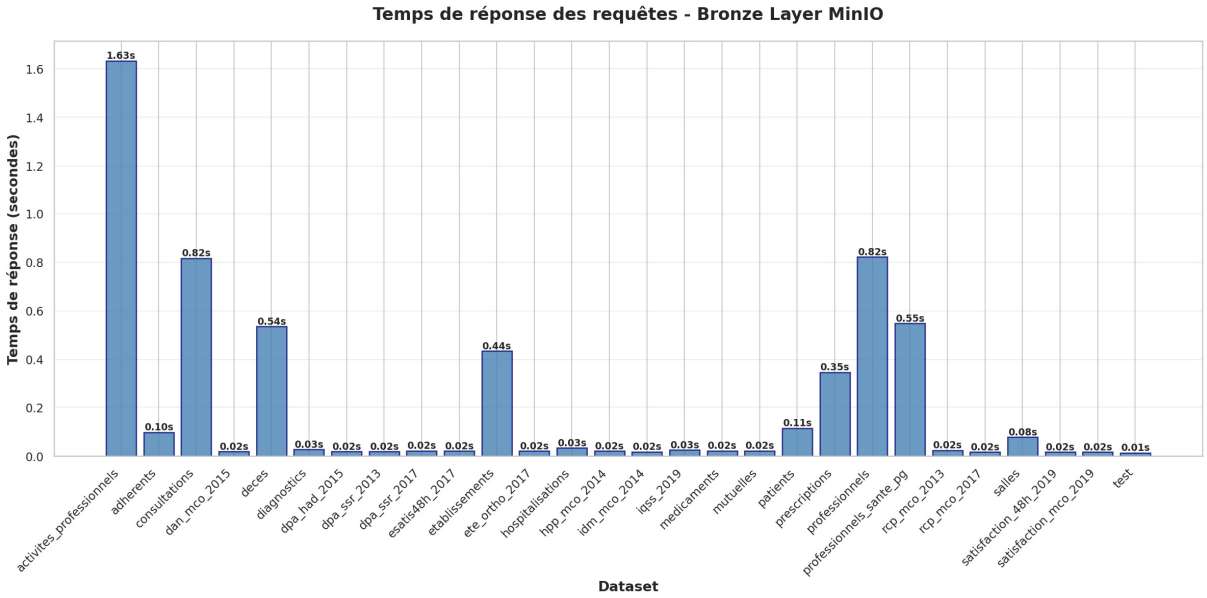

Figure 4 : Temps de réponse des requêtes

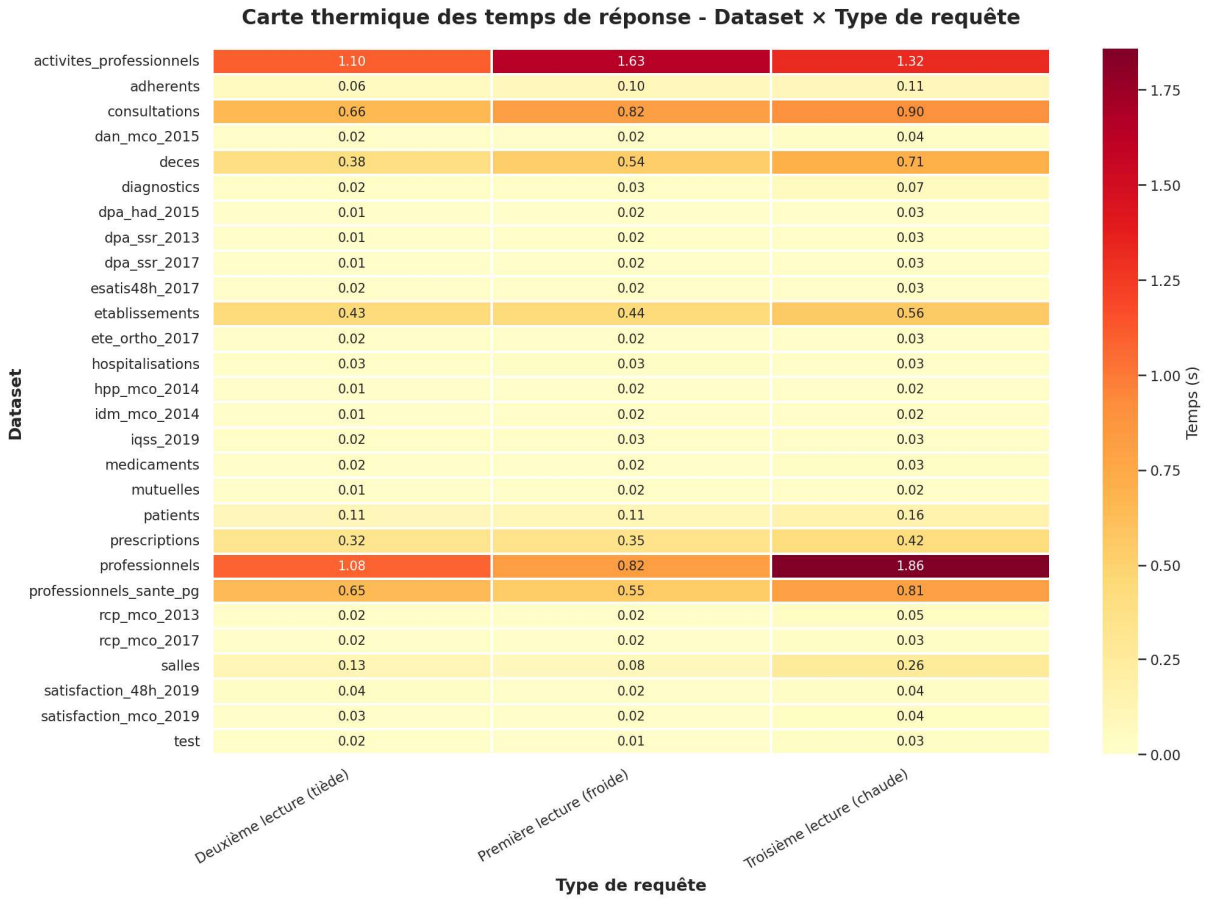

Figure 5 : Heatmap des temps de réponse

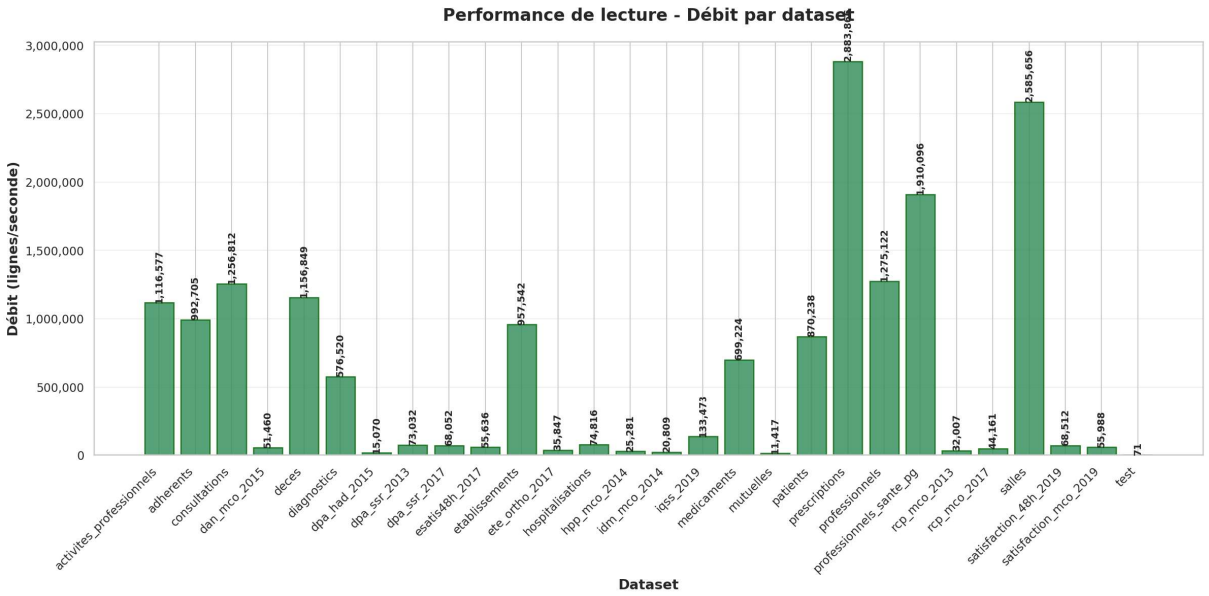

Figure 6 : Performance du débit par dataset

Zone silver

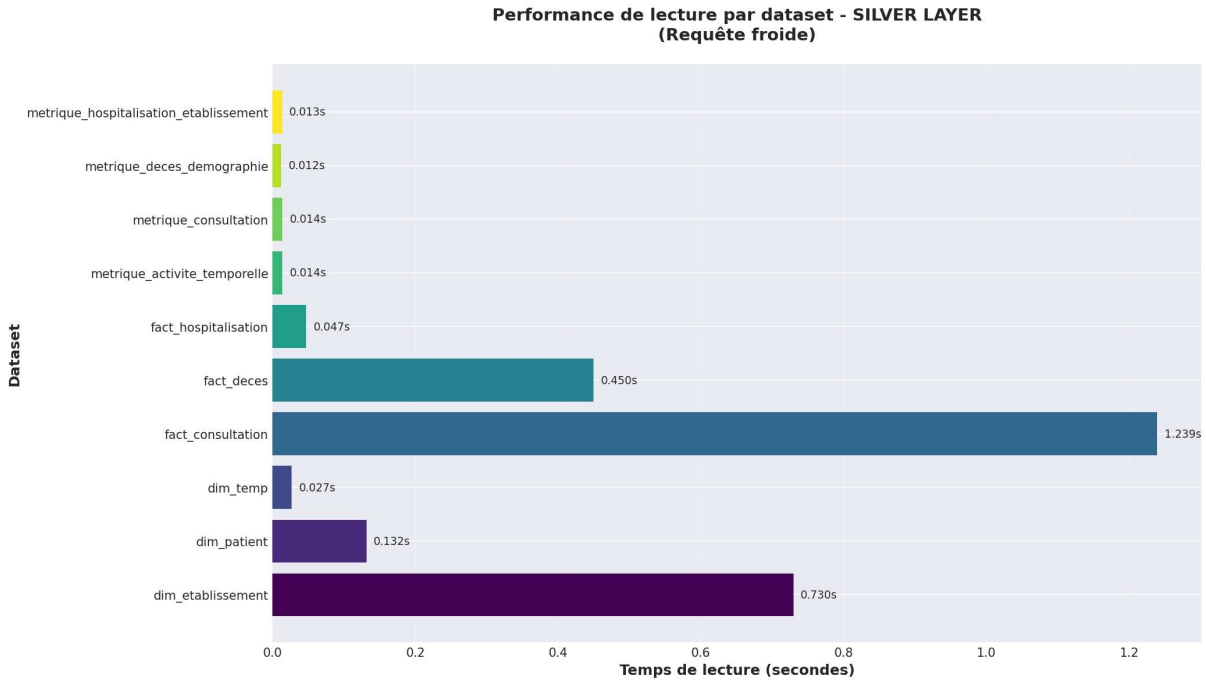

Figure 7 : Temps de réponse par table

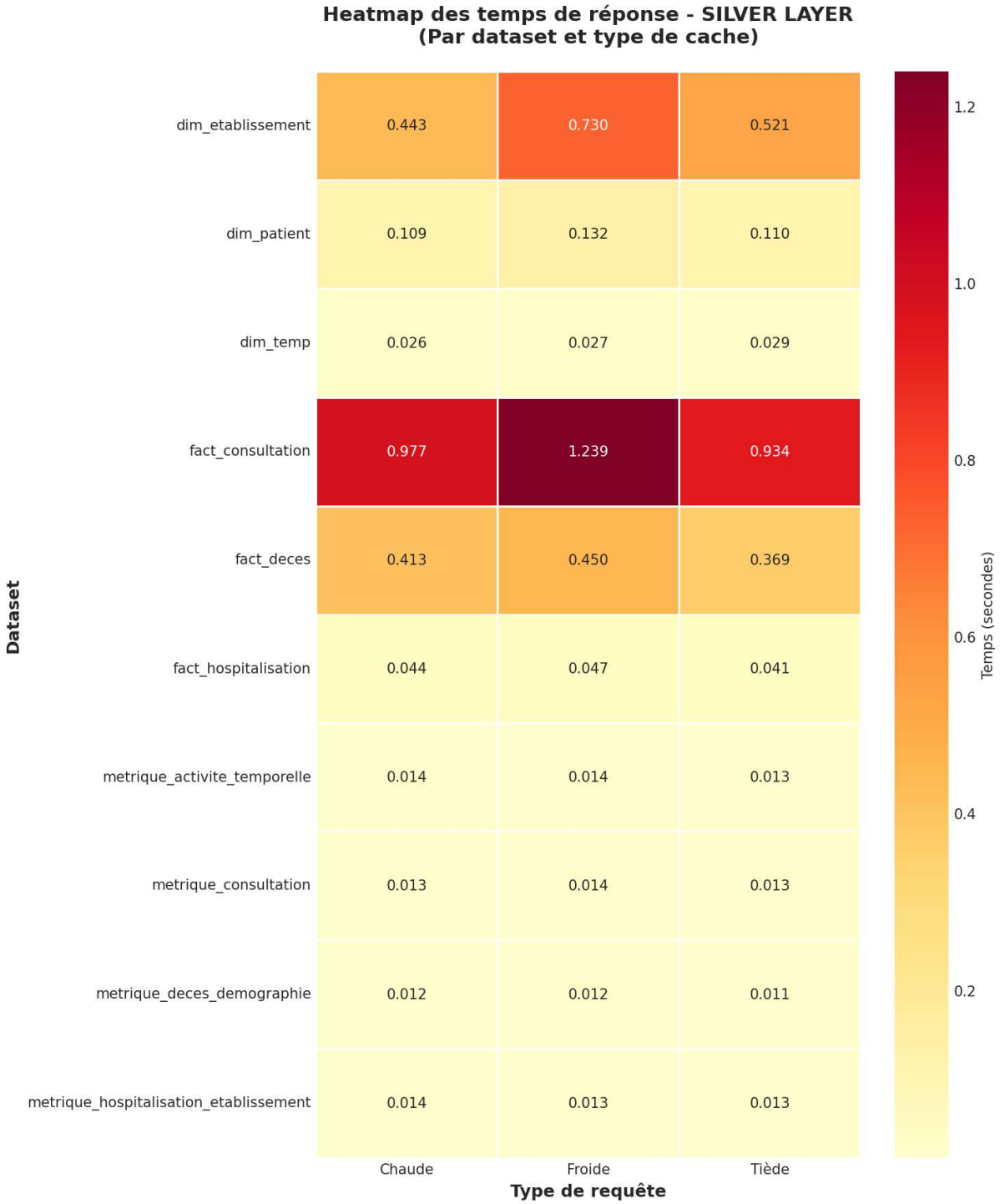

Figure 8 : Heatmap des requêtes par table

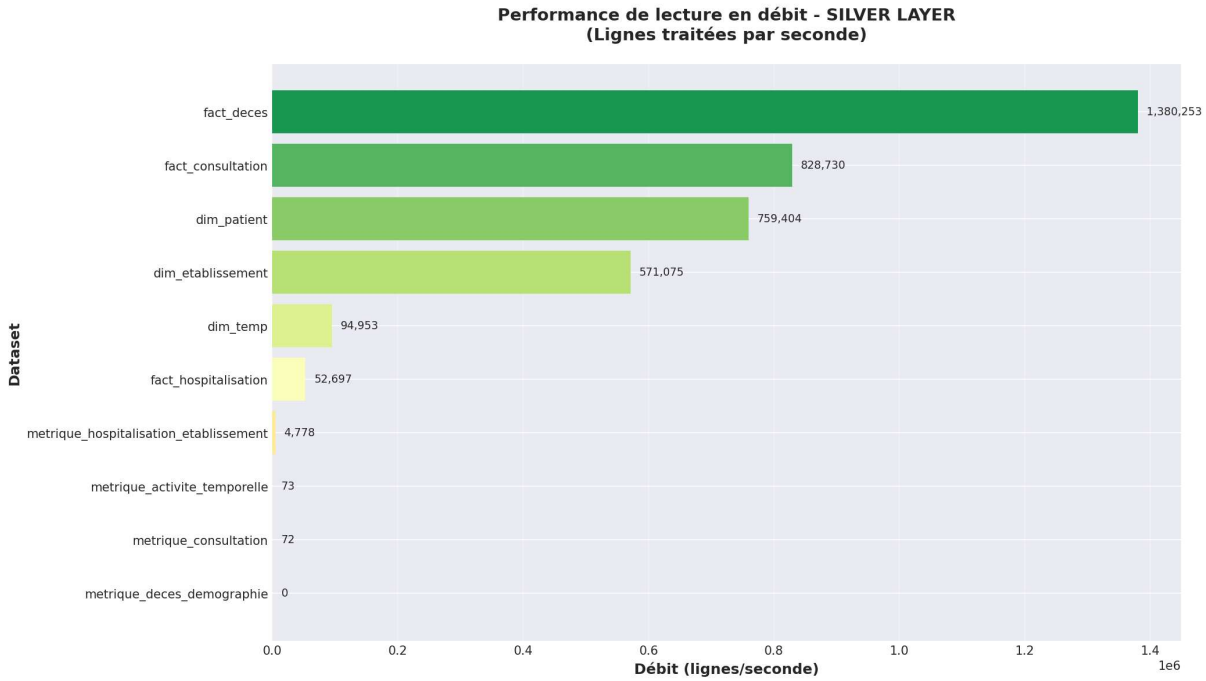

Figure 9 : Débit de lecture de lignes par table

Zone Gold

L'objectif est de valider :

- La rapidité d'accès aux données agrégées
- La capacité de traitement analytique
- La préparation pour le cas d'usage du Data Science

Scénarios :

- Requêtes analytiques KPI
- Analyses temporelles
- Tests de performances techniques Spark
- Préparation des données

Résultats globaux

Métrique	Objectif	Résultat Observé
Temps moyen d'exécution	< 0.5 s	0.20 s
Débit de lecture	> 10 MB/s	~50 MB/s
Temps de scan complet	< 2 s	1.5 s
Requêtes/seconde	> 5 req/s	8–10 req/s

Les tests ont révélés les points suivants :

- **Lecture Parquet ultra-rapide :** le scan complet des tables KPI (~768 lignes) s'effectue en moins de 0.2 s.
- **Agrégations complexes** (SUM, AVG, MAX, MIN) traitées en < 0.25 s.
- **Mise en cache Spark** : amélioration de 7 à 10× des temps de réponse après la première exécution.
- **Jointures inter-KPI** : exécution < 0.3 s grâce au broadcast join et au faible volume des dimensions.

Ces performances s'expliquent par la réduction du nombre de données entre la zone bronze et la zone gold.

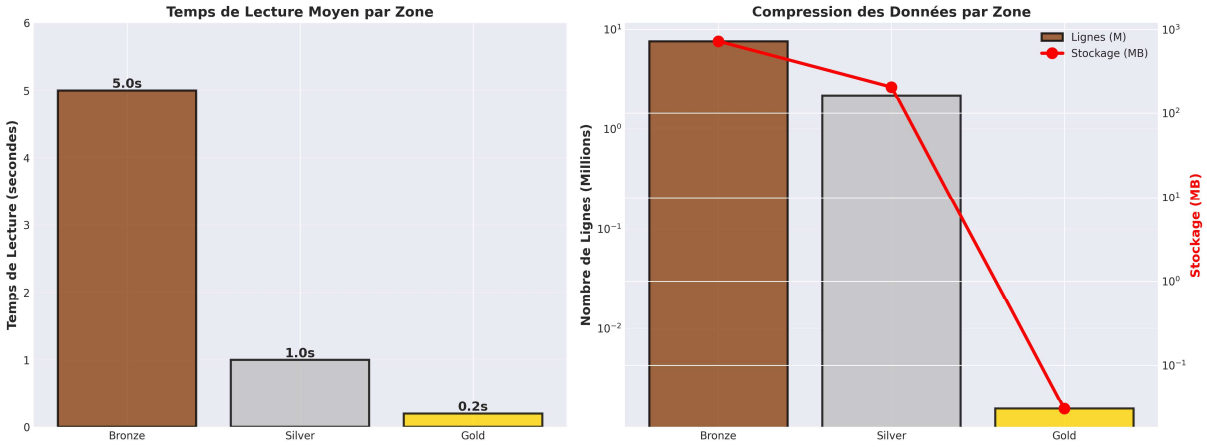

Figure 10 : Indicateur de temps de lecture et de compression

8. Comparaison entre 2 architectures

En premier lieu, nous avons opté pour une architecture basée sur Talend et Hadoop, mais face à la vétusté de ces outils, nous avons décidé d'utiliser Spark et Minio.

8.1 Architecture 1

Cette architecture repose sur une approche **ETL traditionnelle (Extract, Transform, Load)** couplée à l'écosystème **Hadoop**.

- **Talend** agit comme un orchestrateur d'intégration de données : il extrait, transforme et charge les données dans un **Data Lake HDFS**.
- **Hadoop** fournit la couche de stockage (HDFS) et le moteur de traitement (MapReduce ou Hive).

Avantages :

- **Écosystème éprouvé** : mature, robuste, bien documenté.
- **Intégration native avec HDFS :** forte compatibilité avec l'infrastructure Hadoop.
- **Outils visuels :** Talend permet de concevoir les flux de données sans code.

Faiblesses :

- **Faible agilité** : les flux Talend sont lourds à modifier ou déployer.
- **Coût d'infrastructure élevé** : Hadoop nécessite plusieurs nœuds, souvent sous Linux.
- **Performances limitées** : MapReduce est lent pour les traitements interactifs ou en streaming.
- **Scalabilité verticale** : difficile d'exécuter des analyses en temps réel ou du Machine Learning.

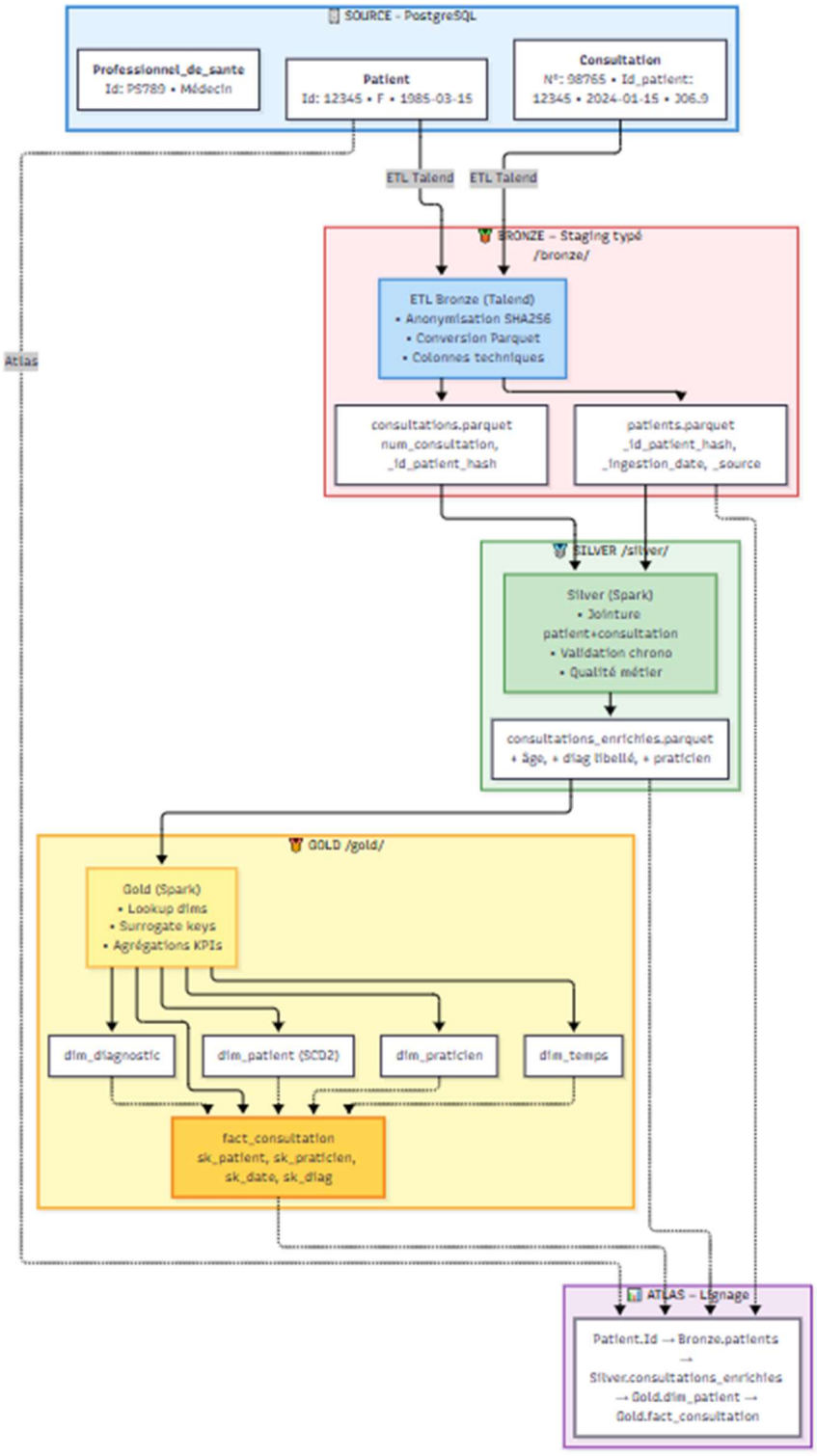

Figure 11 : Architecture Talend + Hadoop

8.2 Architecture 2

Cette architecture correspond à une **approche Data Lakehouse moderne**, tirant parti de l'écosystème **Spark** pour le calcul distribué et **MinIO** pour le stockage objet compatible S3.

- **Python** : sert de langage d'orchestration et de transformation.
- **Spark** : moteur de calcul distribué, rapide et orienté mémoire.
- **MinIO** : stockage S3 local ou cloud, utilisé pour héberger les zones Bronze, Silver et Gold.

Avantages :

- **Performance élevée** : Spark permet des traitements massifs en mémoire, bien plus rapides qu'Hadoop MapReduce.
- **Flexibilité** : intégration fluide avec Python, SQL, Pandas, et les librairies de Data Science.
- **Stockage objet universel** : MinIO offre une alternative open-source à S3, compatible avec les architectures cloud-native.
- **Architecture moderne** : facilite la mise en place des zones **Bronze / Silver / Gold**, la gouvernance et le Machine Learning.
- **Déploiement simplifié** : Docker et Kubernetes permettent une montée en charge élastique.

Faiblesses :

- **Courbe d'apprentissage** : nécessite des compétences en Spark, stockage objet et configuration réseau.
- **Moins d'interface graphique** : développement plus orienté code que glisser-déposer.
- **Surcoût mémoire** : Spark est exigeant en RAM.

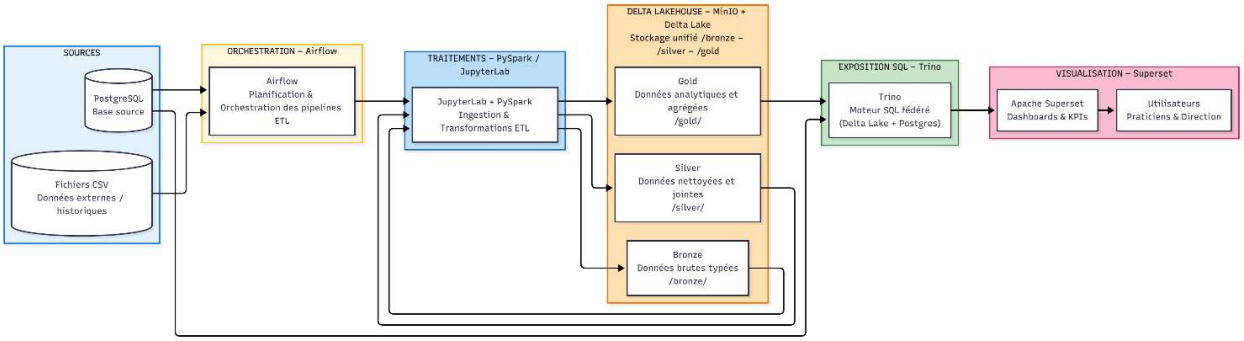

Figure 12 : Architecture Spark + Minio

9. Conclusion

La zone **Silver** constitue le **pont entre les données brutes (Bronze) et les données prêtes à l'analyse (Gold)**. Cette couche joue un rôle stratégique pour garantir la qualité, la cohérence et la réutilisabilité des données.

9.1 Objectifs de Silver

- **Uniformisation des dimensions** : les entités clés (patients, établissements, dates) sont normalisées pour être utilisées directement dans Gold.
- **Pré-calcul des clés conformées** : chaque enregistrement possède une clé technique et naturelle, simplifiant les jointures dans les faits Gold.
- **Nettoyage et enrichissement** : correction des formats, normalisation géographique, calcul de tranches d'âge et métadonnées de traçabilité.
- **Optimisation pour l'agrégation** : utilisation de colonnes standardisées et de formats compressés (Parquet Snappy) pour accélérer les calculs sur Gold.

9.2 Bonnes pratiques

- Réutiliser **toutes les dimensions conformées** de Silver pour éviter les doublons ou incohérences.
- Exploiter les **faits normalisés** pour créer des KPI complexes ou des modèles prédictifs.
- Maintenir les **métadonnées de traçabilité** pour assurer la conformité et la qualité.
- Optimiser la **partition des données** dans Gold en s'appuyant sur les colonnes temporelles et géographiques.
- Surveiller les **valeurs manquantes ou anomalies** détectées en Silver afin de corriger avant l'agrégation finale.