PROSIT : BEE PATIENT

FODIL Nel | MARCELLI Enzo | GOUADFEL Rayan
15 avril 2024

Table des matières

Table des figures

I. Introduction

Dans l'univers dynamique de la santé, où chaque donnée peut sauver des vies, le projet "Bee Patient" émerge comme une solution transformatrice. Nous nous lançons dans la mise en œuvre d'une infrastructure décisionnelle agile avec l'ambition d'améliorer la prise en charge des patients et d'optimiser la gestion des établissements de santé. Cette initiative est le reflet de notre engagement à saisir le potentiel du Big Data pour éclairer les décisions médicales et administratives.

II. Analyse du contexte

Après une rencontre constructive avec les acteurs clés de la santé – médecins, infirmiers, chercheurs, administrateurs –, notre trajectoire est affirmée et nos objectifs sont clairement définis. Les attentes des utilisateurs sont réaffirmées, ancrant notre direction dans la continuité du modèle conceptuel préétabli et les opérations ETL spécifiées. Cette interaction nous propulse vers l'implémentation concrète du modèle physique, tout en nous préparant à embrasser les défis techniques du traitement de données massives via Hive sur Hadoop.

Le cap est mis sur l'efficience avec un processus de mise à jour des données à tous les trimestres, garantissant la fraîcheur et la pertinence des analyses produites. Cependant, la performance est sous les projecteurs avec une attente audacieuse : générer des analyses complexes en moins de deux minutes. Ce défi impose un équilibre délicat entre les capacités de notre architecture et les exigences pragmatiques des utilisateurs.

Nous nous tenons au seuil d'une échéance qui déterminera la viabilité de notre projet. Les jours à venir testeront la rapidité, la précision et la fiabilité de notre système. "Bee Patient" n'est pas seulement un projet ; c'est une promesse de progrès, une course contre la montre pour livrer des analyses qui ne sont pas seulement des chiffres, mais des reflets de vies humaines.

III. Objectifs

- **Validation des Besoins Utilisateurs** : Assurer que les demandes d'analyse recensées lors des réunions précédentes avec les utilisateurs (médecins, infirmiers, chercheurs, administrateurs) restent inchangées et sont bien alignées avec les attentes, sans nécessiter de modifications majeures de la modélisation conceptuelle ou de l'ETL.

- **Implémentation du Modèle Physique** : Passer à la phase d'implémentation physique de la base de données décisionnelle, en remplissant les tables et en préparant les requêtes pour les analyses spécifiées.

- **Optimisation de Performance** : Mesurer l'efficacité de l'utilisation du SGBD NoSQL Hive, notamment en termes de système de partitionnement, pour garantir des temps de réponse acceptables aux requêtes, malgré les grands volumes de données.

- **Mise à Jour des Données** : Établir un processus de mise à jour des données sources tous les trois mois pour maintenir l'actualité des données dans l'entrepôt et pour définir de nouvelles analyses.

- **Respect des Contraintes Temporelles** : Répondre à la contrainte de temps de traitement maximal de 2 minutes pour la sortie des analyses, en exploitant la parallélisation des tâches sur l'environnement Hadoop.

- **Livraison des Analyses** : Réussir à produire les analyses demandées dans les délais impartis pour la phase actuelle du projet, en tenant compte des limites de performance possibles dues au volume des données et à la complexité des requêtes impliquant de multiples jointures.

IV. Problématique

Comment concilier les attentes des utilisateurs en matière de délais de traitement avec la complexité et le volume des données, tout en maintenant la performance des requêtes dans un environnement de données décisionnelles NoSQL tel que Hive sur Hadoop ?

V. Plan d’Action

- Amélioration des requêtes avec Apache Hive :
  Utilisation de Hive pour exploiter sa capacité de traitement SQL sur des ensembles de données de grande taille. La mise en place de tables partitionnées permettra de cibler et de traiter uniquement les segments de données nécessaires pour une requête donnée, optimisant ainsi les temps de réponse.

- Parallélisation et distribution des calculs :
  Exécution de jobs MapReduce pour décomposer les requêtes en sous-tâches exécutées en parallèle. Ceci est essentiel pour gérer efficacement les grandes quantités de données et pour accélérer les temps de traitement.

- Utilisation de vues matérialisées :
  Création de vues matérialisées pour stocker les résultats des requêtes fréquemment utilisées. Cela permettra d'accéder plus rapidement aux données sans recalculer les résultats à chaque fois.

- Mise à jour régulière des données :
  Mise en place d'un calendrier de mise à jour trimestrielle pour assurer la fraîcheur des données dans l'entrepôt. Les processus ETL seront utilisés pour intégrer les nouvelles données et rafraîchir les vues matérialisées.

- Répondre à la contrainte de temps des requêtes :
  Pour respecter la limite des 2 minutes pour le temps de réponse des requêtes, nous concentrerons nos efforts sur l'optimisation des requêtes et sur le partitionnement des données pour minimiser les temps d'accès et de calcul.

- Développement des jobs d'alimentation :
  Création et automatisation des jobs ETL pour le peuplement et la mise à jour de l'entrepôt de données, garantissant l'alignement avec les besoins d'analyse recensés.
  Ces mesures concrètes alignent les capacités techniques avec les besoins exprimés par les utilisateurs, visant à rendre le système décisionnel aussi réactif et performant que possible.

VI. Définitions

Modèle conceptuel des données (MCD)

Le Modèle Conceptuel des Données est une représentation abstraite et structurée des données utilisées dans un système d'information ou une base de données. Il vise à décrire les entités, leurs attributs et les relations entre ces entités dans un langage visuel ou symbolique. Le MCD fournit une vue conceptuelle des données, indépendante de toute considération technique ou de mise en œuvre. Il est souvent utilisé comme point de départ dans la conception d'une base de données.

Schéma décisionnel

Le Schéma Décisionnel, également appelé schéma en étoile ou en flocon, est une structure de base de données conçue pour prendre en charge l'analyse décisionnelle et le data warehousing. Il se caractérise par une table centrale de fait, qui contient les mesures numériques de l'activité commerciale (ventes, quantités, revenus, etc.), entourée de tables de dimension qui décrivent les différents contextes ou dimensions dans lesquels ces mesures sont analysées (temps, produit, lieu, etc.). Le schéma décisionnel facilite les requêtes analytiques complexes et les rapports.

Tables de dimension

Les Tables de Dimension sont des tables dans une base de données ou un entrepôt de données qui contiennent des informations descriptives sur les entités principales dans un schéma décisionnel. Chaque table de dimension représente une perspective ou une dimension spécifique des données, telles que le temps, le produit, le client ou la géographie. Ces tables fournissent des contextes pour analyser les mesures contenues dans la table de fait. Les tables de dimension sont souvent reliées à la table de fait par des clés étrangères.

Tables de fait

Les Tables de Fait sont des tables centrales dans un schéma décisionnel qui contiennent les mesures numériques ou quantitatives de l'activité commerciale, telles que les ventes, les revenus, les quantités vendues, etc. Chaque ligne dans une table de fait représente un enregistrement d'événement ou de transaction, généralement associé à des clés étrangères faisant référence aux tables de dimension. Les tables de fait sont souvent volumineuses et contiennent des données agrégées ou détaillées, selon les besoins analytiques.

Modèle logique de données (MLD)

Le Modèle Logique de Données (MLD) est une représentation plus technique du Modèle Conceptuel de Données (MCD) qui est utilisée pour planifier la structure réelle de la base de données. Le MLD est crucial pour transformer les entités, attributs et relations du modèle conceptuel en une structure prête à être implémentée dans un système de gestion de base de données (SGBD). Voici les aspects clés du MLD :

Tables :

- Chaque entité du modèle conceptuel est convertie en une table.
- Les attributs de l'entité deviennent des colonnes dans la table.

Clés Primaires :

- Chaque table doit avoir une clé primaire qui identifie de manière unique chaque enregistrement (ou ligne) dans la table.
- La clé primaire peut être composée d'un seul attribut ou d'une combinaison de plusieurs attributs (clé composite).

Clés Étrangères :

- Pour représenter les relations entre les tables, des clés étrangères sont utilisées.
- Une clé étrangère dans une table pointe vers une clé primaire dans une autre table, facilitant ainsi l'intégrité référentielle entre elles.

Relations :

- Les relations entre les entités (comme les associations un-à-un, un-à-plusieurs, ou plusieurs-à-plusieurs) sont traduites en utilisant des clés primaires et étrangères.
- Pour les relations plusieurs-à-plusieurs, une table de jointure est souvent nécessaire pour relier les tables concernées.

Normalisation :

- Le processus de normalisation est appliqué pour réduire la redondance des données et améliorer l'intégrité des données. Cela implique la division de grandes tables en plus petites, plus gérables, et l'élimination des dépendances partielles et transitives.

Intégrité des données :

- Des contraintes d'intégrité sont définies pour garantir la validité et l'exactitude des données. Ces contraintes peuvent inclure des contraintes de clé, des contraintes de domaine (type de données, gamme de valeurs), et des contraintes d'intégrité référentielle.

Job d'alimentation (ETL/ELT)

Les jobs d'alimentation en contexte de traitement de données se réfèrent généralement aux processus utilisés pour extraire, transformer et charger (ETL) ou extraire, charger et transformer (ELT) des données d'une source à une destination, typiquement un entrepôt de données ou un lac de données.

ETL (Extract, Transform, Load) :

- **Extract** : Les données sont extraites de leur source, qui peut être des bases de données, des fichiers plats, des flux en direct, ou d'autres systèmes.
- **Transform** : Les données extraites subissent diverses transformations pour assurer leur qualité, leur cohérence et leur pertinence. Les transformations peuvent inclure le nettoyage, la déduplication, l'agrégation, le redimensionnement, le calcul de nouvelles variables, etc.
- **Load** : Les données transformées sont chargées dans la destination finale, souvent un entrepôt de données ou une base de données opérationnelle.

ELT (Extract, Load, Transform) :

- **Extract** : Similaire à ETL, les données sont extraites de leurs sources.
- **Load** : Contrairement à ETL, les données sont chargées directement dans la destination finale, généralement un système capable de gérer de grands volumes de données, comme un lac de données, avant que les transformations ne soient appliquées.
- **Transform** : Les transformations sont réalisées après le chargement des données dans la destination. Ce modèle est particulièrement efficace lorsque la destination a une grande capacité de traitement, permettant de gérer de grands volumes de données plus efficacement.

Architecture de l'entrepôt de données

L'architecture d'un entrepôt de données (data warehouse) décrit la structure et l'organisation des données centralisées utilisées pour l'analyse et la prise de décision au sein d'une entreprise. Cette architecture est généralement divisée en trois niveaux principaux : la couche de base de données où les données sont chargées et stockées, la couche de présentation où les données sont organisées et rendues accessibles aux utilisateurs, et la couche d'accès où les utilisateurs interagissent avec les données à travers des outils d'analyse et de reporting.

- Couche de Source de Données : Inclut les systèmes opérationnels de l'entreprise et d'autres sources externes, où les données sont collectées avant d'être transformées et chargées dans l'entrepôt.
- Zone de Staging (zone d'attente) : Les données sont temporairement stockées ici pour y subir le nettoyage, la transformation et l'intégration avant leur chargement dans la couche de stockage.
- Couche de Stockage de Données : C'est le cœur de l'entrepôt de données, où les données transformées sont stockées. Cette couche peut comprendre plusieurs schémas de modélisation de données, tels que schéma en étoile, schéma en flocon de neige, ou encore des modèles normalisés pour optimiser les requêtes et les analyses.
- Couche de Présentation : Organise les données en cubes ou agrégats qui facilitent l'analyse et les requêtes par les utilisateurs finaux, souvent à l'aide de tableaux de bord et de rapports personnalisés.
- Outils d'Accès aux Données : Incluent des logiciels de Business Intelligence (BI), des outils d'analyse ad hoc, et d'autres applications qui permettent aux utilisateurs de créer des rapports, de visualiser des données, et de réaliser des analyses décisionnelles.

HiveQL

HiveQL est le langage de requête utilisé par Apache Hive, qui est conçu pour interroger et gérer de grandes quantités de données en utilisant une syntaxe similaire à celle du SQL traditionnel. HiveQL permet aux utilisateurs de créer des tables, des vues, et d'interroger des données stockées dans un environnement Hadoop. Le langage supporte des fonctionnalités typiques de SQL telles que les jointures, les groupements, les fonctions d'agrégation, les sous-requêtes, et les requêtes complexes sur les grandes bases de données distribuées. HiveQL est particulièrement puissant pour la transformation de données et l'analyse batch, offrant une méthode accessible pour exécuter des analyses complexes sur de grandes données sans nécessiter une connaissance approfondie de la programmation MapReduce.

Partitionnement Hive

Le partitionnement dans Hive est une technique de gestion des données qui permet de diviser une table en plusieurs parties ou partitions, basées sur les valeurs d'une ou de plusieurs colonnes clés. Cette stratégie est cruciale pour optimiser les performances des requêtes sur de grands volumes de données dans un environnement Hadoop. En partitionnant une table, Hive stocke les données dans différents répertoires au sein du système de fichiers HDFS, correspondant à chaque valeur unique de la colonne de partition. Cette approche permet de limiter le nombre de données lues lors de l'exécution des requêtes qui spécifient une clause WHERE sur la colonne de partition, réduisant ainsi le temps de traitement et améliorant l'efficacité globale du système décisionnel.

Tables partitionnées Hive

Dans Hive, les tables partitionnées sont utilisées pour améliorer l'efficacité des requêtes en stockant les données dans des dossiers séparés basés sur des valeurs de colonnes spécifiques, appelées clés de partition. Chaque partition agit comme une sous-table, permettant à Hive de limiter la quantité de données à lire lors de l'exécution des requêtes qui filtrent par la clé de partition. Cette méthode est particulièrement utile pour les grands ensembles de données distribuées sur un système Hadoop, car elle optimise le traitement en réduisant le temps d'accès et de traitement des données pertinentes. La partition est définie lors de la création de la table en spécifiant une ou plusieurs colonnes de partition, ce qui entraîne le stockage physique des données dans des chemins séparés pour chaque combinaison unique de valeurs de partition.

Optimisation des requêtes Hive

L'optimisation des requêtes Hive est le processus d'amélioration des performances des requêtes exécutées sur Apache Hive, qui est un système de gestion de données sur Hadoop. Cette optimisation peut impliquer plusieurs techniques, telles que la réécriture de requêtes pour une exécution plus efficace, l'utilisation de partitions et de buckets pour minimiser le volume de données lues lors d'une requête, et l'application de vues matérialisées pour pré-calculer et stocker les résultats des requêtes fréquemment utilisées. En outre, le réglage des paramètres de configuration d'Hive et de Hadoop, comme augmenter la mémoire allouée aux tâches ou ajuster le nombre de réducteurs, peut également contribuer à améliorer les performances des requêtes. Ces stratégies visent à réduire le temps de réponse et à accroître l'efficacité de traitement dans des environnements de données volumineuses.

Visualisation des données (Tableaux de bord)

La visualisation des données à travers des tableaux de bord est une méthode permettant de présenter des données complexes de manière graphique et intuitive, facilitant leur compréhension et leur analyse. Les tableaux de bord regroupent et affichent des informations provenant de diverses sources de données sous formes de graphiques, de jauges, de cartes, et d'autres widgets visuels. Cela permet aux utilisateurs de voir rapidement des patterns, des tendances et des anomalies, et de prendre des décisions éclairées basées sur les données en temps réel ou historiques. Les tableaux de bord sont largement utilisés dans les décisions stratégiques, opérationnelles et tactiques, offrant une vue d'ensemble instantanée de la performance, de la santé, et des tendances clés de l'entreprise ou de ses divisions spécifiques.

Gouvernance des données

La gouvernance des données est le processus par lequel une organisation formalise la manière dont elle organise, sécurise, gère et utilise les données pour assurer leur qualité, leur conformité et leur efficacité tout au long de leur cycle de vie. Cela comprend l'établissement de politiques, de procédures, de normes et de responsabilités pour l'administration des données collectées, stockées, utilisées et archivées par une organisation. La gouvernance des données aide à garantir que les données sont utilisées de manière appropriée, qu'elles sont précises et accessibles uniquement aux personnes autorisées, et qu'elles répondent aux exigences réglementaires. Ce processus aide également à réduire les risques de données inexactes ou mal utilisées et augmente la valeur des données en tant qu'actif pour l'organisation.

Performance et scalabilité des requêtes

La performance des requêtes désigne la rapidité avec laquelle un système de base de données peut exécuter des requêtes et retourner les résultats. Elle est essentielle pour garantir que les applications fonctionnent efficacement et que les utilisateurs finaux bénéficient d'une expérience rapide et fluide. La scalabilité des requêtes, quant à elle, fait référence à la capacité du système à maintenir ou à améliorer sa performance au fur et à mesure que le volume de données augmente ou que le nombre de requêtes simultanées croît. Un système bien conçu doit pouvoir gérer des augmentations de charge sans dégradation significative de la performance. Cela implique souvent l'usage de techniques comme le partitionnement de données, l'indexation efficace, et le déploiement de clusters de serveurs qui peuvent traiter des requêtes en parallèle, s'appuyant sur des technologies telles que NoSQL ou des solutions de base de données distribuées pour une meilleure gestion de la montée en charge.

Temps de réponse des systèmes décisionnels

Le temps de réponse dans les systèmes décisionnels fait référence à la durée nécessaire pour que le système récupère et présente les données après une requête de l'utilisateur. Ce facteur est crucial pour l'efficacité des analyses de données et la prise de décision. Un temps de réponse optimal permet aux décideurs d'obtenir des informations pertinentes rapidement, favorisant ainsi des décisions rapides et éclairées. Les temps de réponse peuvent varier en fonction de la complexité des requêtes, du volume de données, de l'architecture du système, et de l'efficacité des mécanismes de traitement et de requête utilisés, tels que les indexations, les optimisations de requêtes et le partitionnement des données.

Conclusion

En conclusion, ce prosit a permis de mettre en lumière les défis techniques et organisationnels associés à la mise en œuvre d'un système décisionnel avancé pour le groupe CHU. À travers une planification méticuleuse et l'adoption de technologies adaptées comme Hive sur Hadoop, nous avons défini des stratégies pour optimiser le traitement et l'analyse des données de santé volumineuses. La mise en place des jobs d'alimentation et l'utilisation efficace du partitionnement dans Hive visent à améliorer significativement les temps de réponse des requêtes, malgré les volumes importants de données à traiter. Cette approche nous rapproche de notre objectif d'offrir des analyses précises et rapides qui sont essentielles pour soutenir la prise de décision au sein du groupe CHU. La suite du projet consistera à affiner ces processus et à valider leur efficacité par des tests rigoureux, en préparation pour la présentation finale et l'évaluation du système complet.

Webographie

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

https://univ.scholarvox.com/reader/docid/88843191/page/1

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

https://hive.apache.org/

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

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

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

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

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

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

https://www.adaltas.com/fr/2018/05/31/accelerating-query-processing-with-materialized-views-in-apache-hive-2/

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

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Joins

https://docs.microsoft.com/fr-fr/azure/hdinsight/hadoop/apache-hadoop-use-hive-ambari-view