République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Saad Dahlab – Blida 1

Faculté des Sciences
Département Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme MASTER EN Informatique

Option : **Ingénierie du Logiciel**

Thème

Data Provenance et Data Lineage dans un environnement Data Lake

Réalisé par :
- AMROUCHE Manel.
- BOUKROU Lina.

Encadré par :
- Mr. M. BALA.

Promotion : 2022/2023
Session : Juin 2023.

Remerciements

Avant toute chose, nous remercions Dieu, le tout-puissant, de nous avoir donné la volonté et la patience de mener à bien ce modeste travail.

Nos remerciements particuliers s'adressent d'abord à notre promoteur, M. M. BALA, pour avoir accepté de nous encadrer, de nous conseiller tout au long de ce parcours de recherche et d'avoir enrichi nos connaissances. Un immense merci pour votre disponibilité, votre gentillesse et votre patience qui nous ont permis de mener à bien ce travail. Vraiment, le travail avec vous était une expérience passionnante.

Nous tenons à exprimer notre profonde reconnaissance aux membres du jury, Mme L.Oukid, présidente, et Mme Tobji, pour nous avoir fait l'honneur de juger ce travail.

À nos chers parents, pour leur soutien inconditionnel tout au long de notre parcours académique. Leur amour, leurs encouragements et leur dévouement ont été des piliers essentiels qui nous ont permis d'atteindre nos objectifs. Leur patience et leur confiance en nous ont été une source d'inspiration constante. Nous sommes profondément reconnaissantes pour tout ce qu'ils ont fait pour nous. Nous espérons avoir été à la hauteur de tout ce qu'ils nous ont apporté.

À nos chère frères nous vous dédiez ce travail en témoignage de l'attachement, l'amour et l'affection que nous portons pour vous, que dieu vous garde et ouvre les portes de la réussite, ainsi que nos chères sœur pour leur soutien moral et leurs sacrifices tous le long de nos études, que dieu vous gardent et vous accordent tous le bonheur et le sucées du monde.

À nos aimables amis qui nous ont apporté beaucoup de joie et de plaisir, nous vous dédions ce travail et nous vous souhaitons une vie pleine de bonheur

Et nous tenons également à remercier tous nos enseignants qui ont participé à notre formation tout au long de notre cursus.

Résumé

La vulgarisation des données à grande échelle, tout particulièrement les données non structurées et les environnements Big Data, a donné naissance à une nouvelle technologie de stockage appelée Lac de données (Data Lake en anglais). Les lacs de données permettent de stocker d'énormes quantités de données, qu'elles soient structurées, semi-structurées ou non structurées, en les publiant à des fins d'analyse de données, de statistiques, de fouille de données, etc.

Pour s'assurer que cette technologie de stockage ne se transforme en marécages, la gouvernance du Data Lake basée sur un catalogue de métadonnées est considérée comme la clé de voûte d'un environnement Data Lake. Le catalogue des métadonnées permet de créer des passerelles sémantiques entres les différentes sources de données et de prendre en charge les différentes opérations de base à savoir l'ingestion des données, faciliter l'accès aux différentes sources quel que soit leur format, etc.

La " Data Provenance " ou " Data Lineage " étant l'un des éléments importants du catalogue des métadonnées du système Data Lake, elle permet de définir le contenu des sources de données, de décrire les liens entre les sources ainsi que l'historique des changements effectués sur celles-ci. L'objectif de ce travail est la capture, le stockage, l'interrogation et la visualisation des métadonnées décrivant le cycle de vie des sources de données dans un environnement Data Lake.

Mots-clés: Data Provenance, Data Lineage, Métadonnées, Data Lake, Big Data, NoSQL.

Abstract

The widespread use of large-scale data, particularly unstructured data, and Big Data environments, has given rise to a new storage technology known as the Data Lake. Data lakes make it possible to store huge quantities of data, whether structured, semi-structured, or unstructured, and publish it for the purposes of data analysis, statistics, data mining, etc.

To ensure that this storage technology does not turn into a swamp, Data Lake governance based on a metadata catalog is the keystone of a Data Lake environment. The metadata catalog makes it possible to create semantic gateways between the various data sources and to support the various basic operations, i.e., data ingestion, facilitating access to the various sources regardless of their format, etc.

"Data Provenance" or "Data Lineage " is one of the key elements in the Data Lake system's metadata catalog. It is used to define the content of data sources, to describe the links between sources and the history of changes made to them. The aim of this work is to capture, store, query and visualize the metadata describing the lifecycle of data sources in a data lake environment.

Keywords: Data Provenance, Data Lineage, Metadata, Data Lake, Big Data, NoSQL.

الملخص

أدى تعميم البيانات واسعة النطاق، وخاصة البيانات غير المهيكلة وبيئات البيانات الضخمة، إلى ظهور تقنية تخزين جديدة تسمى بحيرة البيانات. تتيح بحيرات البيانات تخزين كميات هائلة من البيانات سواء كانت منظمة أو شبه منظمة أو غير منظمة من خلال نشرها لأغراض تحليل البيانات والإحصاءات واستخراج البيانات وما إلى ذلك.

لضمان عدم تحول تقنية التخزين هذه إلى مستنقعات، تعتبر إدارة بحيرة البيانات استنادًا إلى كتالوج البيانات الوصفية حجر الزاوية في بيئة بحيرة البيانات. يتيح كتالوج البيانات الوصفية إمكانية إنشاء بوابات دلالية بين مصادر البيانات المختلفة ودعم العمليات الأساسية المختلفة، أي استيعاب البيانات، وتسهيل الوصول إلى المصادر المختلفة بغض النظر عن تنسيقها، وما إلى ذلك.

يعتبر "مصدر البيانات " او" نسب البيانات" أحد العناصر المهمة في فهرس الميتاداتا لنظام بحيرة البيانات. يجعل من الممكن تحديد محتوى مصادر البيانات، لوصف الروابط بين المصادر بالإضافة إلى محفوظات التغييرات التي تم إجراؤها عليها. الهدف من هذا العمل هو التقاط وتخزين واستعلام وتصور البيانات الوصفية التي تصف دورة حياة مصادر البيانات في بيئة بحيرة البيانات.

الكلمات المفتاحية: مصدر البيانات، نسب البيانات، البيانات الوصفية، بحيرة البيانات، البيانات الضخمة، NoSQL.

Table des matières

Table des Figures

Liste des tableaux

I. Introduction générale

1. Contexte et problématique

La connectivité à l'échelle mondiale à Internet, aux réseaux sociaux, capteurs digitaux, applications e-commerce, etc. a donné naissance à une donnée complexe dans son format, son volume, sa vitesse de génération et de traitement, sa visualisation, etc.

Cette donnée est largement connue aujourd'hui sous le terme « Mégadonnée » (Big Data) et caractérisée par une complexité résumée par les 7Vs à savoir Volume, Vélocité, Véracité, Variété, Visualisation, Valeur, Variabilité. En effet, on parle de volumes allant de centaines de téraoctets, pétaoctets, exaoctets et yottaoctets. Quant à la variété, on distingue des données structurées, semi-structurées et non structurées. La visualisation, quant à elle, implique la complexité de consulter une telle donnée sur les dispositifs d'affichage alors que la véracité exprime une problématique de fiabilité et de confiance vis-à-vis de données provenant de sources diverses non suffisamment identifiées.

Afin d'apprivoiser et de cerner le phénomène 'Big Data', de nombreuses technologies ont vu le jour, permettant aux scientifiques de données de gérer le cycle de vie des données massives comprenant leur création, stockage, modifications et traitements. Cette thématique devient aujourd'hui plus importante dans des environnements Data Lakes connus par leur capacité à stocker de grandes quantités de données structurées, semi-structurées et non structurées en préservant leur format natif.

Mais bien que la technologie Data Lake ait fait ces preuves face aux enjeux du Big Data, elle présente toutefois une multitude de défis lors de son utilisation, plus particulièrement ceux liés à sa gouvernance qui inclut l'ensemble de pratiques visant à contrôler les Data Lake, cela comprend notamment la gestion des métadonnées (données sur les données).

En effet, la capacité du Data Lake à ingérer de grandes quantités et tous formats de données sans schémas préalables (approche schemaless) constitue sa force mais aussi un défi dans le sens où cela peut éventuellement se transformer en un marécage de données (ensemble de données sans stratégie de stockage) sans l'intégration d'un catalogue de métadonnées. Ceci montre clairement l'avantage des catalogues de métadonnées abritant des objets, catégories, propriétés et champs puisqu'ils apportent au Data Lake l'enrichissement sémantique en rendant les données stockées plus interprétables, compréhensibles et déterminent les passerelles sémantiques entre celles-ci. Ils contribuent

également à l'indexation des données pour faciliter l'accès et l'interrogation des données en plus du versionnement de celles-ci et suivi de leur utilisation. Parmi tous les aspects traités par un catalogue de métadonnées dans un Data Lake, les informations relatives au cycle de vie des données (création et manipulation) relèvent du Data Lineage qu'on peut également qualifier de cartographie du cycle de vie des données du Data Lake ou provenance de données vu qu'il définit l'historique des changements des données que ce soit au niveau de la structure, schémas, paramètres d'accès, etc. En outre, le système de data provenance permet la capture des opérations opérées par différents utilisateurs sur les données.

2. Objectifs et contributions

Ce PFE objective de proposer une approche de data provenance couvrant essentiellement les phases de capture, de stockage, d'interrogation et enfin de visualisation des métadonnées relatives au cycle de vie des sources du Data Lake. Afin de mettre en œuvre cette approche, nous envisageons de développer un prototype qui prend en charge les différents aspects proposés ci-dessus.

Notre application DATA PROVENANCE, fonctionnant comme un service en arrière-plan, permet d'extraire automatiquement les métadonnées sur la provenance pendant l'exécution des différents traitements opérés sur les sources du Data Lake. Notre plateforme comportera essentiellement les modules suivants :

1. Module de capture de provenance.

2. Module d'ingestion des sources de données dans le Data Lake.

2. Module de stockage distribué des métadonnées.

3. Module de visualisation.

Notre système prend en charge trois formats de données à savoir documents PDF, documents WORD et vidéo MP4.

3. Organisation du mémoire

Afin de bien structurer notre travail, notre mémoire s'organise selon l'ordre suivant :

Partie 1 : se constitue des trois premiers chapitres servant à définir les concepts de base liés à notre problématique. Le premier chapitre introduit le Big Data. Le deuxième chapitre est dédié aux lacs de données et enfin le troisième chapitre est attribué à la Data Provenance et aux concepts qui y sont relatifs.

Partie 2 : se compose de deux chapitres. Dans le quatrième chapitre, nous exposons quelques travaux connexes à notre travail ainsi qu'une comparaison entre ces derniers. Nous introduisons dans le cinquième chapitre notre proposition dans le cadre de ce PFE.

Partie 3 : Conçue en un chapitre, nous décrivons dans cette partie notre environnement de travail et les technologies utilisées pour concrétiser notre plateforme. Pour finir, nous expliquons le fonctionnement de notre plateforme et ses différents usages

I. Big Data

Ce chapitre présente les concepts relatifs au big Data et tout particulièrement les 5Vs, les formats de données, l'environnement Big Data le plus populaire au monde à savoir Apache Hadoop et enfin les modèles de données NoSQL.

---

1. Introduction

Nous assistons actuellement à une forte augmentation de la quantité d'informations suite à l'utilisation à grande échelle d'Internet, smartphones, réseaux sociaux et autres technologies. En effet, en 2011, le volume global de données créées et copiées dans le monde était de 1,8 ZB (≈ 10^{12}B) [1]. Il devient nécessaire de convertir ces données en informations faciles à utiliser, d'où l'apparition du terme « Big Data » qui comprend généralement des masses de données à grande échelle non structurées provenant de différentes sources.

2. Définition du Big Data

Bien que le volume ne soit pas réellement suffisant à lui seul, le big data est le terme souvent utilisé dans la désignation de données avec des volumes inhabituels, des formats très divers générés en des temps records avec une grande vitesse. La complexité des mégadonnées est généralement caractérisée par les cinq dimensions connues sous le nom des « 5 Vs ».

En 2011, le McKinsey Global Institute propose la définition suivante : « Le Big Data se réfère à un ensemble de données dont la taille va au-delà de la capacité des logiciels de bases de données classiques à capturer, stocker, gérer et analyser ». [2] (traduction)

En 2013, le terme « Big Data » entre officiellement dans l'Oxford English Dictionary. Il est défini comme « des données d'une très grande taille, dans la mesure où leur manipulation et leur gestion entraînent d'importants défis logistiques » [3] (traduction).

3. Caractéristiques du Big Data

Il y a cinq caractéristiques principales, dans la littérature on les appelle les « 5 Vs » (voir la figure 2.1) : variété, valeur, volume, véracité, vitesse (vélocité).

Volume : Il fait référence à la quantité massive de données générées, collectées et stockées. Le Big Data implique la manipulation de vastes ensembles de données, souvent de l'ordre du téraoctet, du pétaoctet ou même de l'exaoctet.

La vitesse : les données sont générées de manière rapide et doivent donc être traitées rapidement pour extraire des informations pertinentes, afin d'en faire le meilleur usage, sinon elles perdent leur utilité. La vitesse désigne alors la fréquence à laquelle les données sont générées, capturées, stockées, partagées.

La variété : les Big Data se démarquent par leurs formats variés, elles sont générées à partir de différentes sources, et ont, par conséquent, diverses structures (documents textes,

vidéos, images…). Elles peuvent aussi être complètes ou incomplètes, privées ou publiques, partagées ou confidentielles.

La valeur : Big Data est synonyme de big valeur, en effet, pour que l'utilisateur puisse utiliser l'information de la meilleure façon, celle-ci doit avoir une réelle signification pour l'utilisateur et de l'intérêt après son utilisation, autrement dit, en termes d'analyse

La véracité : on se focalise par ce critère sur l'aspect qualitatif des données car pour en profiter, on doit bien avant tout s'assurer de l'intégralité de ces dernières, plus clairement, non endommagées par le processus de décryptage. Il est finalement question de juger la crédibilité et la fiabilité des données collectées.

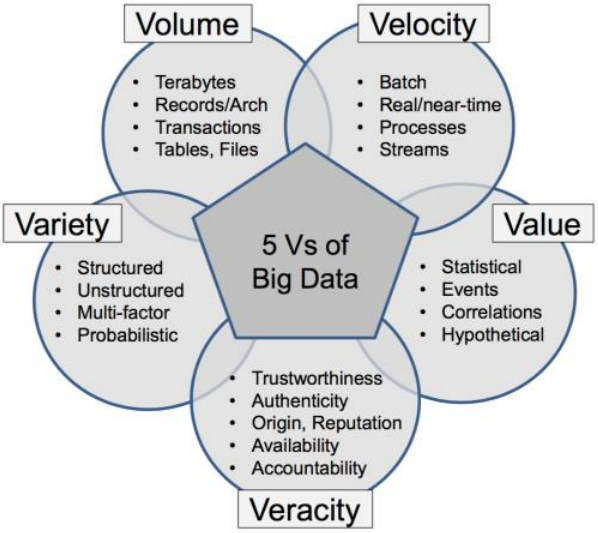


Figure 2.1: Les 5 Vs du Big Data [4].

4. Classification du Big Data

Les mégadonnées peuvent se présenter sous diverses formes : messages, appels, courriels, commandes en ligne, photos et musique partagées, etc. Elles peuvent être classifiées selon trois types :

Non structurées : des données ne possédant pas un format permettant d'y accéder et nécessitant donc un traitement humain, étant donné l'impossibilité de leur trouver une structure prédéfinie. On reconnaît : les documents textes, multimédias, vidéos, CV, etc.

Structurées : en opposition aux données non structurées, celles-ci ont une structure prédéfinie pouvant être traitée, organisée, manipulée automatiquement par la machine. Elles sont généralement stockées dans des bases de données relationnelles et interrogées à l'aide de SQL. Parmi ces données, on a : données GPS, factures, etc.

Semi-structurées : représente une forme intermédiaire. Il est souvent compliqué de classer un document dans l'une des catégories précédentes, par exemple : en ajoutant des métadonnées tels que des mots clés définissant son contenu à un document Word (non structuré), on obtient alors une donnée semi-structurée.

5. Technologies du Big Data

Les technologies Big Data sont désormais très variées dans leurs applications et leurs Framework, tels que le Cloud Computing, la plateforme Apache Hadoop, l'internet des objets IoT, le NoSQL, ayant pour objectif principal leur utilisation dans d'autres domaines tels que le Machine Learning, l'intelligence artificielle, le Deep Learning et l'analyse prévisionnelle. Parmi ces technologies, deux sont considérées comme plus importantes : **Hadoop** et **NoSQL**.

5.1. HADOOP

Un Framework open source servant à stocker et traiter des données de très gros volumes en mode non-relationnel et distribué. Il peut être considéré comme étant un écosystème se composant principalement de :

- **Hadoop HDFS** : couche de stockage de données (système de fichiers).
- **Hadoop MapReduce** : le moteur de traitement dans Hadoop s'exécutant sur plusieurs clusters selon deux fonctions Map et Reduce.
- **Hadoop YARN** : le gestionnaire de ressources dans Hadoop.

5.2. NoSQL

Un nouveau type de SGBD différent du modèle relationnel qui s'est avéré inefficace devant la volumétrie et la vélocité du phénomène big data « Not only SQL », proposant quatre modèles d'orientations différentes puisqu'ils sont sans schéma (schemaless), contrairement aux bases de données relationnelles avec un schéma prédéfini, à savoir : le modèle orienté clé-valeur, graphe, document, colonne.

- **Modèle orienté clé-valeur** : son principe est de lier chaque valeur dans la base de données (de n'importe quel type) à une clé de type caractère uniquement.
- **Modèle orienté colonne** : proche du modèle relationnel ayant un nombre déterminé de colonnes, ce modèle est plus flexible et permet l'ajout d'autres colonnes par la suite.
- **Modèle orienté document** : ressemble au modèle clé-valeur, sauf que la valeur est un document de format non imposé. Les données y sont stockées à l'intérieur et peuvent différer selon les documents.
- **Modèle orienté graphe** : indépendant des autres modèles, il est composé de nœuds représentant les données et les arcs qui correspondent aux relations entre celles-ci.

6. Conclusion

Le big data est sans doute l'un des phénomènes les plus importants dans le monde de la technologie, dirigeant les ingénieurs et autres vers d'autres pratiques que nous avons détaillées plus haut, telles que le NoSQL, dont l'utilisation permet de stocker et traiter les données d'une grande masse et de différents types.

II. Data Lake et Métadonnées

Dans ce chapitre, nous aborderons la définition du Data Lake et la terminologie liée à ce concept telle que l'architecture, les caractéristiques, les technologies, et pour finir, nous aborderons la notion des métadonnées au sein du Data Lake.

---

1. Introduction

Les volumes de données dans le big data, leur nature diverse et non structurée, nécessitent une solution plus flexible que l'entrepôt de données qui est rigide et limité en tant que cadre de gestion des données massives et variées. La solution à ces limites est donc le Data Lake.

2. Définition du Data Lake

Un Data Lake est une solution de stockage et de gestion des big data qui peut stocker une grande quantité de données dans leur format natif (structuré, semi-structuré et non structuré), sans imposer un schéma au niveau de l'ingestion et ne les traiter qu'en cas de besoin. Au sein des Data Lakes, le traitement de ces données est fastidieux, long et inefficace en raison des différents modèles de données et formats de fichiers. De plus, l'accès à ces données est entravé par la variété des interfaces, des services et des applications.

Ce terme est apparu pour la première fois en 2010 par James Dixon l'ayant défini comme suit : « Si vous considérez un Data Mart comme un magasin d'eau en bouteille - nettoyé, emballé et structuré pour une consommation facile - le lac de données est une grande masse d'eau dans un état plus naturel. Le contenu du lac de données s'écoule d'une source pour remplir le lac, et divers utilisateurs du lac peuvent venir examiner, plonger ou prélever des échantillons. »

3. Caractéristiques du Data Lake

Une Data Lake doit inclure ces caractéristiques :

- Un catalogue de métadonnées qui renforce la qualité des données.
- Politiques et outils de gouvernance des données.
- Accessibilité à différents types d'utilisateurs.
- Intégration de tout type de données.
- Une organisation logique et physique.
- Évolutivité en termes de stockage et de traitement.

4. Architecture du Data Lake

La figure 3.1 décrit l'architecture d'un Data Lake. Le sous-système d'acquisition permet de collecter les données à n'importe quelle latence, le deuxième sous-système est le système de gestion de données intervenant dans l'intégration et les processus effectués sur les données (tris, agrégations, jointures, etc.), le dernier sous-système permet d'accéder aux applications choisies par les utilisateurs du Data Lake (BI, DATA MINING, etc.)

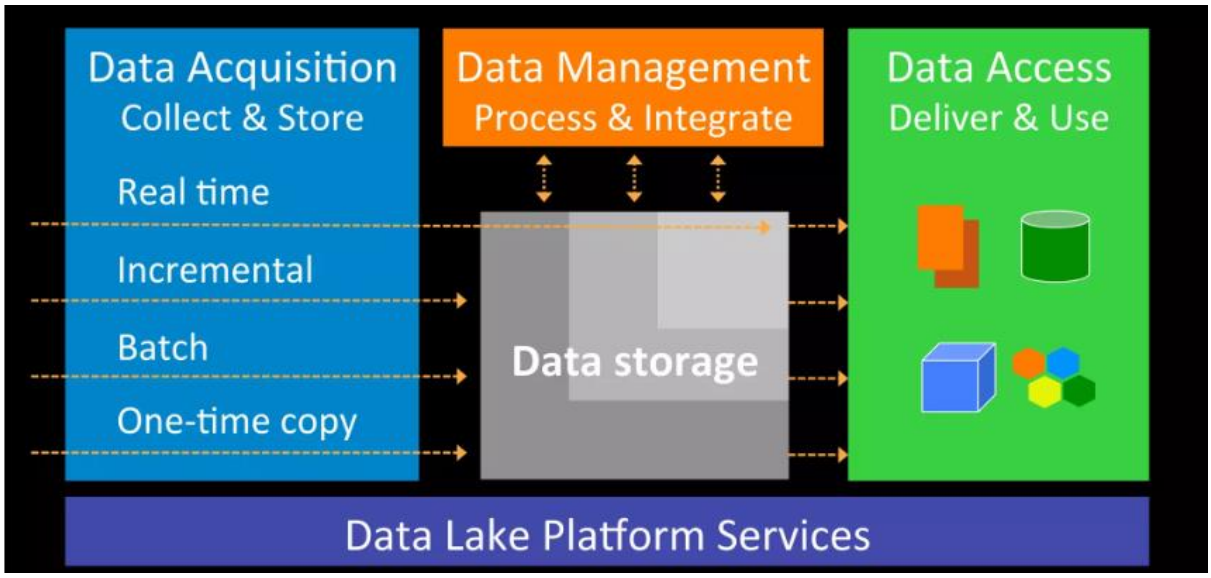


Figure 3.1: Architecture d'un Data Lake [10].

5. Technologies pour le Data Lake

La plupart des implémentations de lacs de données sont basées sur l'écosystème Apache Hadoop, qui a en effet l'avantage de fournir à la fois du stockage avec HDFS et des outils de traitement de données via MapReduce ou Spark. Cependant, il existe d'autres outils utilisables à cet effet, comme indiqué dans la figure 3.2:

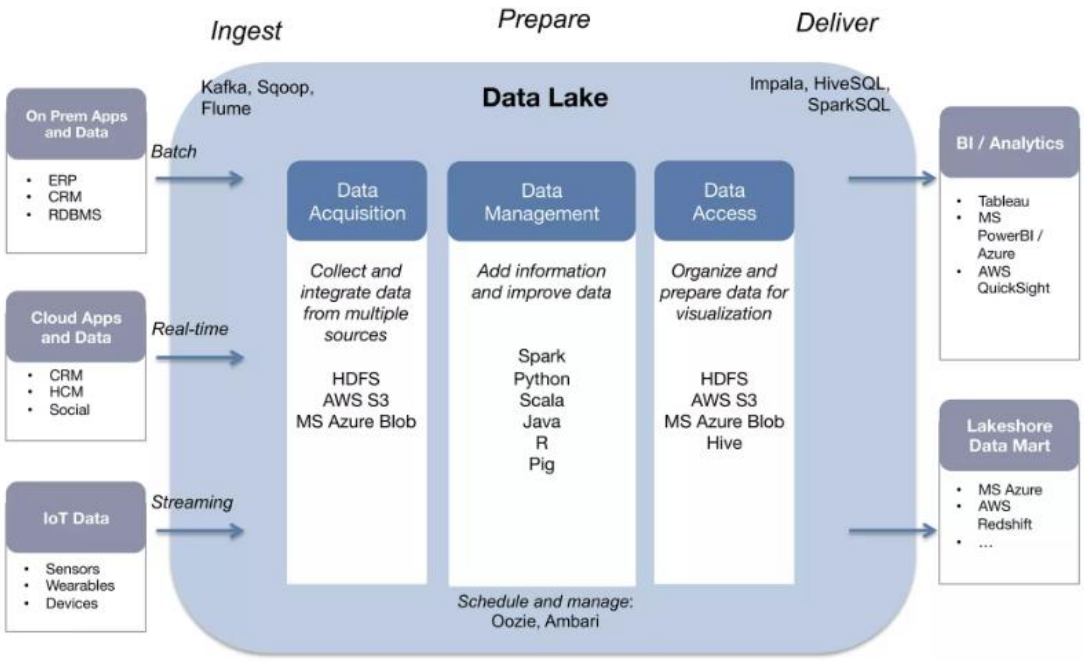


Figure 3.2: Technologies pour les Data Lake [10].

5.1. Ingestion de données

Il existe deux modes d'ingestion, dont l'ingestion manuelle et automatique, qui est plus courante au sein des grandes entreprises. Elle se fait par les technologies d'ingestion suivantes qui permettent de transférer physiquement les données des sources vers un lac de données : Flink, Samza, Kafka et Sqoop.

5.2. Stockage des données

Environ 75 % des Data Lakes [5] utilisent le stockage HDFS. Cependant, il y a certains Data Lakes utilisant des SGBD relationnels ou NoSQL

5.3. Traitement des données

Le Data Lake est également un espace de traitement des données. On peut effectuer différents prétraitements via de nombreux langages de programmation. Voici quelques-uns des langages les plus populaires :

Python : est un langage de programmation très populaire, offrant de nombreuses bibliothèques dédiées au traitement de données comme Pandas, NumPy, Matplotlib et Scikit-learn. Il est utilisé pour le développement de scripts de traitement de données en batch ainsi que pour l'analyse de données en temps réel avec des Framework tels qu'Apache Spark.

R : est un langage de programmation open source spécialement conçu pour l'analyse de données et la visualisation."

Java : est un langage de programmation largement utilisé dans le développement d'applications de traitement de données distribuées, en collaboration avec des Framework populaires tels qu'Apache Hadoop, Apache Spark et Apache Flink.

Scala : est un langage de programmation polyvalent qui se distingue par sa capacité à développer des systèmes de traitement de données distribuées. Il est largement utilisé en combinaison avec Apache Spark pour effectuer des opérations de traitement en temps réel sur des données distribuées.

SQL (Structured Query Language) : est un langage de requête utilisé pour interroger des bases de données relationnelles, ainsi que pour interroger des données stockées dans un Data Lake en utilisant des outils tels que Presto, Apache Drill et Amazon Athena.

Il est important de noter qu'il existe de nombreux autres langages de programmation qui peuvent être utilisés pour le traitement de données dans un Data Lake, et le choix dépendra des besoins spécifiques du projet et des préférences des développeurs.

5.4. Accès aux données

Les technologies d'accès aux données d'un Data Lake varient en fonction d'architecture et des outils utilisés pour stocker les données, tels que chaque base de données dispose d'outils ou de requêtes qui nous permettent d'accéder aux données qui y sont stockées et de les traiter.

6. Gestion des métadonnées dans les Data Lakes

La nature du "dump everything" dans les Data Lakes et l'intégration des données sans schéma explicite augmente la flexibilité du stockage de données, mais en même temps peut le transformer en marécage de données en l'absence d'un système de métadonnées efficace.

6.1. Définition de métadonnée

Dans son acception première, le terme signifie « données sur les données ou données qui renseignent sur certaines données et qui permettent ainsi leur utilisation pertinente » (Vocabulaire de la géomatique publié par l'Office de la langue française en 1993). Les métadonnées sont définies dans [6] comme « les informations relatives aux données collectées elles-mêmes et contiennent des informations concernant la structure et la sémantique des données ». De plus, les métadonnées ne sont pas là par accident : elles ont été explicitement ajoutées (manuellement ou automatiquement) pour aider quelqu'un, quelque part, à un moment donné, à trouver les informations associées, afin de faciliter la localisation des objets. Elles fournissent des informations sur chaque ensemble de données, telles que la taille, le schéma d'une base de données, le format, l'heure de la dernière modification, les listes de contrôle d'accès, l'utilisation, etc. [9].

6.2. Catégories de métadonnées

Dans la littérature, ils classent les métadonnées selon deux catégories de catégorisation :

6.2.1. Métadonnées fonctionnelles

La catégorisation proposée par Oram catégorise les métadonnées en fonction de la manière dont elles sont collectées.

- **Métadonnées métier (business) :** sont définies comme l'ensemble des descriptions établies par les utilisateurs professionnels au stade de l'ingestion des données [10].
- **Métadonnées opérationnelles** : sont des informations générées automatiquement lors du traitement des données. Elles capturent la lignée, la qualité, le profil et la provenance [9].
- **Métadonnées techniques :** comprennent des informations sur la forme et la structure de chaque ensemble de données, telles que la taille et la structure du schéma ou le type de données (texte, images, JSON, etc.). La structure du schéma comprend les noms des champs, leurs types de données, leurs longueurs, s'ils peuvent être vides, etc.

6.2.2 Métadonnées structurelles

Catégorisation proposée par Sawadogo et al., qui catégorisent les métadonnées en fonction des "objets" auxquels elles se rapportent.

- **Les métadonnées intra-objet :** appartiennent à un ensemble de caractéristiques associées à des objets uniques dans le Data Lake [8].
- **Les métadonnées inter-objets :** représentent des liens entre deux ou plusieurs objets [8]. Les métadonnées globales sont des structures de données qui fournissent une couche de contexte pour faciliter le traitement et l'analyse des données. Les métadonnées globales ne sont pas directement associées à un objet spécifique, mais concernent potentiellement l'ensemble du lac [8].

7. Conclusion

Comme nous l'avons expliqué, les Data Lakes représentent une technologie très efficace face aux big data, en permettant leur stockage et leur traitement de manière efficace et centralisée. Cependant, Ils peuvent également présenter des défis en termes de gouvernance des données, d'où l'importance des différents types de métadonnées.

III. DATA PROVENANCE

Dans ce chapitre, nous explorerons les concepts clés liés à la provenance des données, les techniques et les outils utilisés pour capturer, stocker et gérer les informations de provenance, ainsi que les défis et les opportunités associés à la gestion de la provenance des données.

---

1. Introduction

Les quantités de données ainsi que leurs formats augmentent de plus en plus, faisant de leur gestion un réel défi. La gestion de plus en plus complexe implique l'utilisation de métadonnées riches et descriptives pour accompagner les données afin de les comprendre et de les réutiliser dans les organisations, dans un environnement Data Lake. Parmi celles-ci, on a la provenance des données, un type de métadonnées concernant l'historique de dérivation d'un produit de donnée à partir de ses sources d'origine.

2. Définition Data provenance

La provenance des données est définie en divers termes selon l'endroit où elle est appliquée, en général c'est l'information sur les activités, les entités et les personnes impliquées dans la production d'un produit de données au cours de son cycle de vie. D'après [11], le terme provenance est couramment utilisé comme synonyme du mot "lignage" pour désigner la source ou les origines d'un objet ou d'un élément de données. Plus précisément, la provenance fait référence à la quantité totale d'informations comprenant tous les éléments et leurs relations qui contribuent à l'existence d'une donnée. Ainsi, la connaissance de la provenance comprend non seulement des aspects tels que les sources et les étapes de traitement, mais aussi des dépendances et des informations contextuelles. La data provenance a donc pour principal objectif de comprendre l'origine des données dans les bases de données.

Bien que la majorité de la littérature ait définit la « Data Provenance » et le « Data Lineage » comme étant un seul concept, il y a cependant, un bon nombre de data scientistes, qui estime que la Data Provenance se concentre sur l'origine et le contexte général des données, tandis que le lineage des données se focalise sur les relations et les dépendances entre les données dans un système ou un processus.

3. Application de la provenance

La provenance des données est importante pour de nombreuses raisons tel que :

- **La qualité des données :** La provenance est importante pour garantir la qualité des produits. Elle peut être utilisée pour estimer la qualité et la fiabilité des données en fonction des sources de données et des transformations [12]. Elle peut également fournir des déclarations de preuve sur la dérivation des données [13].
- **Le diagnostic d'erreur et le débogage :** La provenance des données peut être utilisée pour détecter les attaques informatiques et les fraudes. En vérifiant la provenance

des données, il est possible de détecter les activités suspectes et d'identifier les sources des attaques.

- **La piste d'audit :** la provenance peut être utilisée pour tracer la piste d'audit des données [15], déterminer l'utilisation des ressources et détecter les erreurs dans la génération de données [16].
- **L'attribution :** Pedigree peut établir le droit d'auteur et la propriété des données, permettre leur citation [14] et déterminer la responsabilité en cas de données erronées.
- **La reproduction de processus :** la provenance n'est pas seulement une trace de ce qui a été fait, mais aussi un moyen de le reproduire. Pour une expérience scientifique, par exemple, les détails sur les conditions de son déroulement, les instruments utilisés, les logiciels utilisés ainsi que leurs versions doivent être enregistrés pour pouvoir reproduire l'expérience et aboutir aux mêmes résultats que l'expérience initiale [18].
- **La prise de décision :** la provenance des données peut être utilisée pour la prise de décision et l'optimisation des processus. En fournissant des informations sur l'origine et l'historique des données, il est possible de comprendre leur contexte et de prendre des décisions éclairées.

Tout bien considéré, La provenance joue un rôle essentiel dans divers domaines, assurant ainsi la transparence, la qualité, la sécurité et la traçabilité des données et des produits.

4. Importance de la provenance dans un Data Lake

Dans un Data Lake, le cycle de vie d'une donnée comprend différentes étapes telles que l'acquisition, l'ingestion, le stockage, la transformation, l'analyse et la publication des données. À chaque étape, des métadonnées peuvent être capturées pour aider à comprendre l'historique et la provenance des données. Voici quelques exemples de métadonnées de provenance qui peuvent être capturées à différentes étapes du cycle de vie d'une donnée dans un Data Lake

- **Métadonnées d'acquisition :** Ces métadonnées peuvent inclure des informations sur la source des données, telles que le nom de l'application ou du système à partir duquel les données ont été extraites, le nom du fichier source, l'heure et la date d'acquisition, ainsi que les informations sur le fournisseur et le contrat.
- **Métadonnées d'ingestion :** Ces métadonnées peuvent inclure des informations sur la façon dont les données ont été ingérées dans le Data Lake, telles que le mode d'ingestion (manuelle ou automatique), le type de connexion, le protocole utilisé, la fréquence d'ingestion, etc.
- **Métadonnées de stockage :** Ces métadonnées peuvent inclure des informations sur l'emplacement de stockage des données dans le Data Lake, telles que le nom du bucket, le type de stockage, les politiques de rétention, les options de sécurité, etc.
- **Métadonnées de transformation :** Ces métadonnées peuvent inclure des informations sur les transformations appliquées aux données, telles que le nom des scripts ou des programmes de transformation, le type de transformation, les paramètres de transformation, etc.

- **Métadonnées d'utilisateurs :** également connues sous le nom de données d'utilisateur ou de profils d'utilisateur, font référence aux informations associées à un utilisateur dans un système informatique ou une plateforme en ligne. Ces métadonnées fournissent des détails sur l'identité, les préférences et les activités d'un utilisateur.
- **Métadonnées de dispositif d`accès :** Ces métadonnées peuvent inclure des informations sur la façon dont les données ont été publiées ou partagées avec d'autres utilisateurs, telles que le nom du service de publication, les permissions d'accès, les informations sur l'audit, etc.

5. Capture de provenance

Toute application de provenance est munie de méthodes de capture de provenance qui ont différentes propriétés, soit les types de transformations qu'elles prennent en charge, la manière dont elles capturent la provenance et le moment où la provenance est capturée. Toutes les méthodes ont quatre aspects :

5.1. Le moment où la provenance est capturée

- **Paresseux (Lazy** en anglais**) :** la provenance d'un produit de données est calculée uniquement lorsque cela est nécessaire, en outre, rétroactivement. Généralement utilisées par les systèmes de bases de données, elles ont pour désavantage qu'elles ne sont pas toujours possibles.
- **Hâtive (Eager** en anglais**) :** la capture de provenance se fait au même moment que les transformations, c'est-à-dire de manière proactive. Ces méthodes sont utilisées par les systèmes de workflows. Elles ont pour désavantage leur coût élevé.

5.2. Techniques de capture

Pour calculer la provenance, nous pouvons soit partir du résultat d'une transformation, puis tracer ces résultats en remontant jusqu'aux entrées dont ils dépendent, ou nous pouvons partir des données d'entrée et suivre leur impact sur les résultats.

- **Annotation :** c'est le traçage vers l'avant.
- **Inversion :** c'est le traçage vers l'arrière.

5.3. Mécanisme de capture

Si une structure interne ou un service externe au système de provenance est utilisé pour capturer les données, en particulier, la stratégie externe est adoptée pour collecter les métadonnées de provenance à la fois des environnements distribués et hétérogènes [17].

5.4. Niveaux de capture

Fait référence aux niveaux de capture, c'est-à-dire aux niveaux où les formes distinctes de provenance peuvent être rassemblées tel que workflow, base de données, entrepôt de données, système d'exploitation OS, activité.

6. Stockage des métadonnées de provenance

Les informations de provenance peuvent être volumineuses par rapport aux données elles-mêmes. On parle alors de provenance finement détaillée. La manière dont elles sont

stockées, c'est-à-dire le lieu et le format de stockage, influe donc sur leur scalabilité, leur coût et leur capacité à être utilisées. Elles peuvent être stockées à l'intérieur comme à l'extérieur du système de capture. Il existe plusieurs techniques de stockage de provenance, comme la compression de provenance et le stockage sous forme de graphe ou d'arbre, ainsi que ces systèmes :

- Bases de données de provenance : Les bases de données de provenance sont des systèmes qui permettent de stocker les informations sur l'origine et l'historique des données et des processus. Ces systèmes peuvent être utilisés pour enregistrer des informations sur les sources de données, les transformations appliquées aux données, les auteurs des données, etc.
- Systèmes de fichiers de provenance : Les systèmes de fichiers de provenance permettent de stocker des informations sur les fichiers, les répertoires et les changements apportés aux fichiers. Ces systèmes sont utiles pour suivre l'historique des fichiers et pour comprendre les modifications apportées aux fichiers.
- Provenance enregistrée dans les métadonnées : Les métadonnées sont des informations structurées qui décrivent les caractéristiques des données. Les métadonnées peuvent être utilisées pour enregistrer des informations sur la provenance des données, telles que les auteurs, les sources, les transformations appliquées, etc.
- Systèmes de blockchain : Les systèmes de blockchain sont des systèmes de stockage décentralisés qui permettent de stocker des informations de manière sécurisée et transparente. La blockchain peut être utilisée pour enregistrer des informations sur la provenance des données, ce qui garantit l'authenticité et l'intégrité des informations enregistrées.

En conclusion Il n'y a pas de solution unique pour le stockage de la provenance, et il est important de choisir le système de stockage en fonction des besoins spécifiques de chaque application. Le choix du système de stockage dépendra des exigences en matière de sécurité, de performances, de fiabilité et de coût, entre autres facteurs.

7. Interrogation et visualisation des métadonnées de provenance

La capture et le stockage seraient inutiles sans l'option d'interrogation des données de provenance ainsi que leur visualisation interactive. Cela permet, en effet, d'en tirer profit pour différents usages. L'interrogation se fait via un processus exploratoire, soit avec des langages de requêtage pas forcément dédiés à la provenance, soit avec un langage fourni par l'application de provenance. Pour la visualisation et le partage, l'utilisateur peut demander l'accès aux informations globales, mais aussi à des parties précises de la provenance stockée, avec un graphe de dérivation. Les API de récupération de provenance peuvent permettre aux utilisateurs de mettre en œuvre leur propre mécanisme d'utilisation.

8. Challenges de la provenance avec l'apparition du Big Data et Data Lakes

Le Big Data et l'apparition des Data Lakes ont largement influencé le domaine de la recherche en matière de Data Provenance puisque :

- Les informations de provenance capturées peuvent être plus larges que les données qui sont elles-mêmes qualifiées de Big Data, elles doivent soit être stockées de manière efficace, soit être réduites afin de ne pas réduire leur disponibilité.
- Le coût pour la gestion de la provenance, surtout pour le Big Data, car elles sont généralement de nature distribuée.
- Il est difficile de stocker et d'intégrer la provenance distribuée.
- Il est difficile de reproduire une exécution à partir de la provenance pour les applications Big Data. De nombreux systèmes de provenance existants n'enregistrent que les données intermédiaires générées pendant l'exécution et leurs dépendances. L'information sur l'environnement d'exécution, également importante pour la reproductibilité, est souvent négligée.

9. Conclusion

En conclusion, les métadonnées de provenance peuvent fournir des informations précieuses sur l'historique et la traçabilité des données dans un Data Lake de manière efficace et sécurisée, facilitant la gestion, la qualité et la sécurité des données, le suivi de leurs évolutions et leurs contextes, ainsi que la prise de décisions éclairées.

V. Travaux connexes

Ce chapitre présente une synthèse de la littérature portant sur la gestion de la Data Provenance. Ce chapitre est structuré comme suit : une introduction, la présentation des approches, ainsi qu'un tableau comparatif entre celles-ci basé sur plusieurs critères pour finir avec une discussion.

---

1. Introduction

Nous assistons ces dernières années à une avancée rapide des solutions de provenance et d'approches très diverses telles que RAMP, NEWT, KARMA, KEPLER, TITIAN, KOMADU... etc.

Dans ce qui suit, nous détaillons quatre approches, à savoir : Data Lake Catalog web application [19], Komadu [20, 21, 22], Kepler [23], Titian [24, 25].

Pour discuter des différents aspects de la provenance des données, il est essentiel de parler de certains critères tels que :

- La technique de capture de métadonnées de provenance.
- Le stockage des métadonnées de provenance.
- La visualisation des métadonnées de provenance.

2. Approche de H. Dibowski et al [19]. (2020)

H. Dibowski et al. proposent une nouvelle architecture pour les Data Lakes. Ils soulignent que la simple centralisation et le stockage des données ne suffisent pas à résoudre les problèmes critiques de gestion des données, tels que la recherche, l'accessibilité, l'interopérabilité et la réutilisation. Ainsi, ils introduisent une couche sémantique dans l'architecture qui apporte une description significative des ressources du Data Lake. Cette description est représentée sous forme d'un graphe de connaissances et inclut des informations sur le contenu des ressources, leur provenance et les autorisations de contrôle d'accès. Pour incorporer cette couche sémantique, ils utilisent l'ontologie DCPAC (catalogue de données, provenance et contrôle d'accès) représentée dans la figure 4.1, qui fusionne plusieurs vocabulaires d'ontologies couramment utilisés et les aligne entre eux.

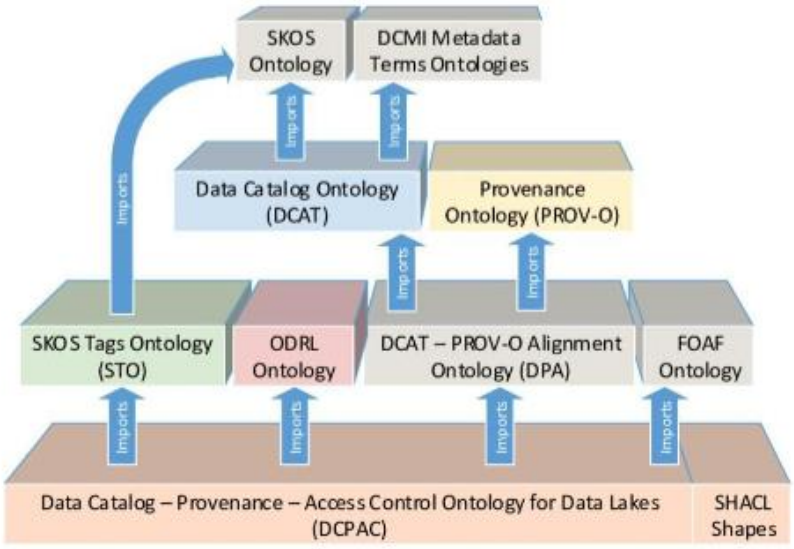


Figure 4.1: Architecture en couches de l'ontologie du catalogue de données, de provenance et du contrôle d'accès (DCPAC) pour les Data Lake.

DCAT : est un vocabulaire RDF faisant une description sémantique du contenu des ressources du Data Lake.

PROV-O : ontologie de provenance dont l'objectif est de décrire la provenance des ressources du DL (comment créées, par quelle activité et quel agent, à partir de quelles données).

ORDL : l'ontologie Open Digital Rights Language appliquée à la définition des autorisations de contrôle d'accès pour les ressources du Data Lake, y compris qui peut accéder à une ressource et quelle action est autorisée. DPA : permet l'alignement entre DCAT et PROV-O pour exprimer des informations de provenance avancée.

SKOS (STO) : l'ontologie des balises utilisées pour spécifier la sémantique des ressources dans un lac de données, en fonction de leur sujet. SHACL Formule : c'est une fiche associée à l'ontologie DCPAC pour la validation des données.

DCPAC : la principale contribution à l'architecture, elle combine, aligne et étend les vocabulaires de l'ontologie de la couche sémantique."

1.1. Architecture du Data Lake et rôle du Semantic Data Lake Catalog

Ils ont construit un Automotive Data Lake en tant que plateforme pour gérer la complexité et l'énorme volume de données. Ils ont créé une couche sémantique 'Semantic Data Lake Catalog' qui fournit une description sémantique des ressources dans le DL et comprend un graphe de connaissances qui est construit avec le vocabulaire défini dans le DCPAC. Les composants de cette architecture sont :

1.1.1. PROCESSUS D'INGESTION DE DONNÉES

Le processus d'ingestion est responsable de l'extraction, de la transformation et du chargement de nouvelles données dans les 'data stores'. À ce stade, il encapsule les données ingérées avec une liste d'opérations standard pour signaler les informations de processus en

tant que données d'entrée et de sortie vers un cluster KAFKA. Ces rapports publiés sous forme de messages KAFKA standard sont consommés par le serveur de population du catalogue du DATA LAKE.

1.1.2. SERVEUR DE POPULATION DU CATALOGUE DU DATA LAKE

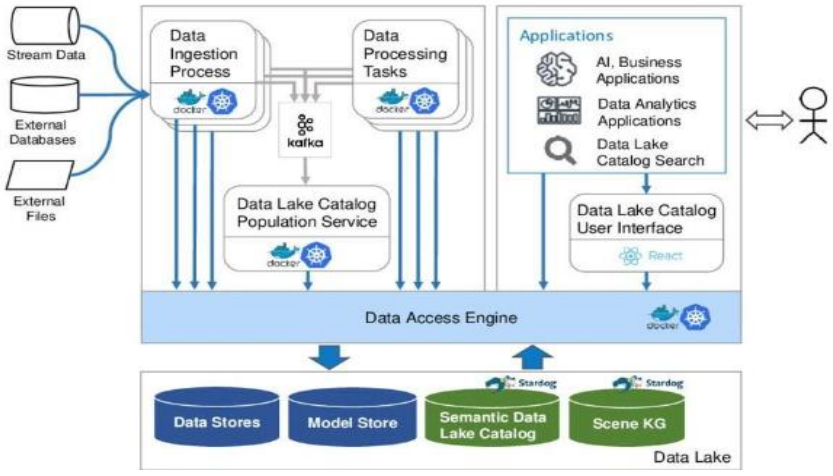
- Lit les métadonnées disponibles sur les données ingérées et construit les données sémantiques pertinentes comme entrée pour le Catalogue du Data Lake Sémantique.
- Aligne, annote et enrichit les données d'entrée du processus d'ingestion de données avec les concepts DCPAC avant de remplir le Catalogue du Data Lake Sémantique.

1.1.3. DATA ACCESS ENGINE(DAE)

- Fournit aux applications une interface de requête uniforme et un accès aux ressources dans le Data Lake sur la base d'une API et d'un point de terminaison HTTP communs.
- Utilise le Semantic Data Lake Catalog pour contrôler l'accès aux ressources de données individuelles sur la base d'opérations d'accès courantes.
- Permet aux développeurs frontaux - sans connaissance de RDF et SPARQL - d'interroger les données pertinentes pour leurs applications.
- Permet aux administrateurs et aux ingénieurs de données autorisés d'interroger le lac de données et le catalogue de lac de données sémantique avec des requêtes SPARQL natives.

1.2. Interface utilisateur du 'Semantic Data Lake Catalog'

Représente une application web développée à l'aide du Framework React qui permet aux administrateurs de lac de données et aux ingénieurs de données de gérer et rechercher des ressources de données disponibles dans le lac de données (voir figure 4.2)



4.2: Architecture du Data Lake et rôle du Semantic Data Lake Catalog.

3. Contribution

Cette approche a été créée en vue de gérer les quatre principes de la « FAIR data » : Faciles à trouver, Accessibles, Interopérables, Réutilisables, en ajoutant une couche sémantique au Data Lake qui est une ontologie et un graphe de connaissances contenant la provenance. Les principales contributions apportées à travers cet outil sont :

- La création d'une ontologie DCPAC (catalogue de données, provenance, contrôle d'accès).
- Le développement du graphe de connaissances correspondant à la précédente ontologie.
- L'implémentation de l'application.

4. Limites

Cette approche présente tout aussi bien des limites que nous citons :

- Elle n'a pas pris en compte toutes les exigences de provenance.
- Elle ne fournit pas une représentation graphique des données de provenance.

3.Approche de I. Suriarachchi et B. Plale [20, 21, 22]. (2016)

I. Suriarachchi & B. Plale proposent une architecture de référence pour la gestion de la Data Provenance dans un Data Lake (voir figure 4.3), basée sur un sous-système de provenance central 'KOMADU', qui stocke et traite les événements de provenance qui y sont pompés à partir de tous les systèmes connectés et font une évaluation de cette architecture à l'aide d'un prototype d'implémentation.

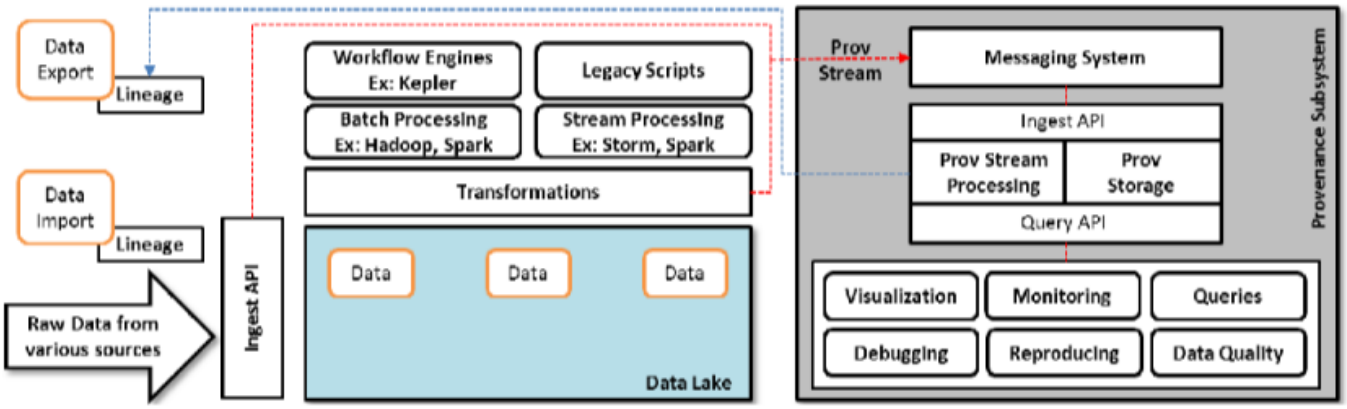


Figure 4.3: Architecture de référence pour la provenance dans un Data Lake.

Le prototype DL a été implémenté à l'aide d'un cluster HDFS et les transformations ont été effectuées avec Hadoop et Spark. Ils ont mis en place une chaîne de transformations basée sur les données Twitter pour d'abord compter les balises d'hashtag, puis obtenir des décomptes agrégés en fonction des catégories. Ces données subissent plusieurs transformations telles que :

- Utiliser Apache Flume pour collecter les données Twitter et les stocker dans HDFS via l'API de diffusion publique Twitter. Pour chaque tweet, Flume capture le pseudonyme Twitter de l'auteur, l'heure, la langue et le message complet, puis les stocke dans un fichier HDFS.
- Après avoir collecté des données Twitter sur une période de cinq jours, une tâche Hadoop a été utilisée pour compter les balises d'hashtag dans l'ensemble des données Twitter et les stocker dans un nouveau fichier HDFS.
- Une tâche Spark utilise ces données (stockées dans le nouveau fichier HDFS) pour obtenir des comptes agrégés selon des catégories prédéfinies.

Toutes ces tâches ont capturé des événements de provenance. Chaque événement se présente sous la forme d'une relation entre deux éléments (par exemple : activité-activité, activité-entité, etc.) et contient des identifiants d'éléments. Ces événements sont diffusés dans le sous-système de provenance où ils sont traités, stockés et analysés.

3.1. Capture de métadonnées de provenance

Pour chaque transformation, les scientifiques des données qui écrivent le code de traitement des données peuvent instrumenter leur code afin de générer la provenance à toutes les étapes nécessaires.

- Le code de capture des tweets dans Flume a été instrumenté pour capturer la provenance des données insérées dans le Data Lake, notamment le pseudo Twitter de l'auteur, l'heure, la langue et le message complet.
- Les fonctions Map et Reduce dans la tâche Hadoop, ainsi que les fonctions MapToPair et ReduceByKey dans la tâche Spark, ont été instrumentées pour capturer la provenance des transformations.

La provenance est généralement représentée sous la forme d'un graphe acyclique orienté (G = (V, E)). Un nœud (V) peut représenter une activité, une entité ou un agent, tandis qu'une arête (E) représente une relation entre deux nœuds. Dans le modèle de collecte de provenance, un événement de provenance est toujours représenté par une arête dans le graphe de provenance. Lorsque tous les systèmes connectés au Data Lake continuent d'envoyer des événements de provenance au système central de collecte de provenance, cela permet de créer un graphe de provenance complet.

3.2. Sous système de provenance

KOMADU est un Framework de collecte de provenance basé sur W3C PROV destiné pour gérer l'ingestion de tous les types de notifications des composants distribués via les canaux de messagerie et les services Web RabbitMQ4.

Les événements de provenance sont générés par Flume, Hadoop et Spark et traités de manière asynchrone par KOMADU puis stockés dans des tables relationnelles (MySQL). La génération de graphes se fait sur demande de visualisation.

Ce système possède deux APIs, une destinée pour l'ingestion des sources du Data Lake et les notifications de provenance alors que l'autre est dédiée pour l'interrogation les informations de provenance.

3.3. Visualisation des métadonnées de provenance

Après avoir exécuté les travaux Hadoop et Spark, l'API de requête KOMADU a été utilisée pour générer des graphiques de provenance. Komadu génère des graphiques de provenance au format PROV XML et est livré avec un plugin Cytoscape qui permet de les visualiser et de les explorer.

3.4. Contribution

Il existe plusieurs techniques pour collecter la provenance à partir de cadres de traitement de Big Data, mais la plupart sont couplées à un cadre particulier et utilisent leurs propres méthodes de stockage ou des normes différentes. La provenance de tous les sous-systèmes doit être assemblée pour créer une trace de provenance plus approfondie. Cependant, même s'il existe des techniques d'assemblage, elles introduisent beaucoup de frais généraux.

Comme solution à ces défis, KOMADU propose quatre contributions. Tout d'abord, il identifie les problèmes de gestion et de traçabilité des données dans un lac de données qui peuvent être résolus en utilisant la provenance. Ensuite, il met en évidence les défis liés à la capture de la provenance intégrée. Troisièmement, il propose une architecture de référence pour surmonter ces défis. Enfin, il évalue la viabilité de l'architecture proposée en utilisant une implémentation prototype avec des techniques qui peuvent être utilisées pour réduire les frais généraux de capture de provenance. En outre, la quatrième contribution consiste à fournir des informations de provenance des données et des processus.

3.5. Limites

Cependant, le système KOMADU possède quelques failles dont :

- Il stocke les données de provenance dans une base de données relationnelle 'MySQL'.
- Il ne prend en charge que les requêtes sur la provenance stockée et ne dispose pas d'un traitement en temps réel des flux de provenance.
- Il ne prend pas en charge les exigences non fonctionnelles telles que la sécurité.

4. Approche de J. Wang et al [23]. (2011)

J. Wang et al. Présentent une architecture pour capturer et interroger la provenance des tâches "map" et "reduce" exécutant des sous-workflows Kepler. L'évolutivité de la collecte et de l'interrogation de la provenance est évaluée en utilisant l'application WordCount et une application bio-informatique appelée BLAST.

4.1. ARCHITECTURE PROPOSÉE :

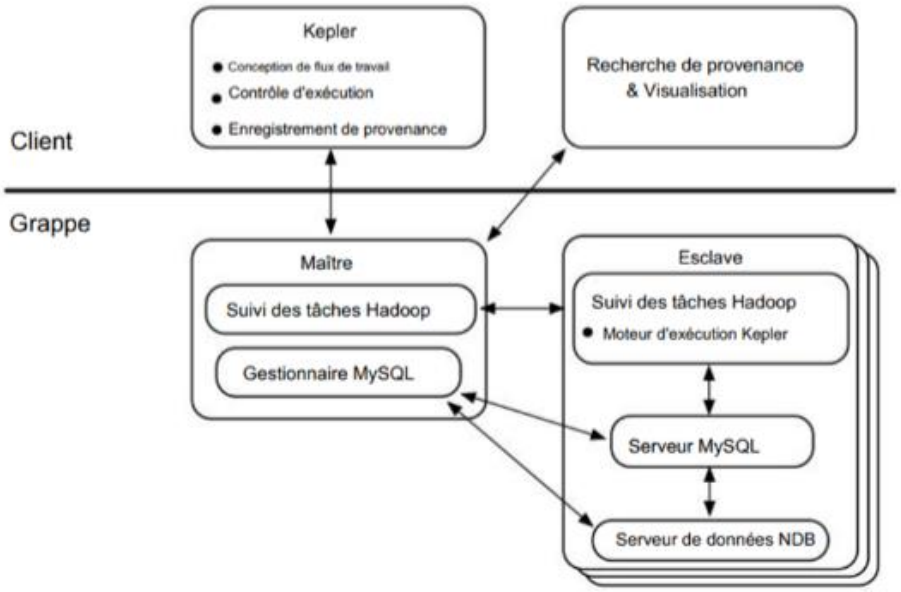


Figure 4.4: Architecture pour la capture de Data Provenance pour des taches MapReduce.

Comme le montre la figure 4.4, cette architecture est composée de deux couches : la première est une interface client pour interroger et visualiser les informations de provenance, et la deuxième est utilisée pour la capture et l'enregistrement de la provenance distribuée à l'aide de MySQL Cluster. Elle a été étendue pour inclure l'enregistrement de provenance distribuée (MySQL Cluster) en étendant l'API d'enregistrement de la provenance et en ajoutant une interface client autre que celle de Kepler (une application de bureau).

La communication entre ces composants se fait à travers des jetons et des messages définis par le réalisateur. Grâce à l'interface graphique de Kepler, il est possible de concevoir des workflows MapReduce et de contrôler leur exécution. Le traitement MapReduce s'exécute en premier lieu sur des données stockées sur HDFS. Pendant l'exécution des sous-workflows de mappage et de réduction, l'API d'enregistrement enregistre les données et les dépendances entre les acteurs.

Un identifiant est attribué à chaque donnée distribuée par le nœud émetteur et utilisé pour alerter le système de provenance par le nœud récepteur.

4.2. Capture et stockage de métadonnées de provenance

Pour stocker les valeurs de toutes les données transmises entre les acteurs lors de l'exécution du workflow, ainsi que les dépendances entre les données et les acteurs, ils utilisent un modèle de données pour décrire les entités et les relations capturées. Ils

considèrent toutes les données et les événements comme des artefacts A et les actions comme des processus P, et ils sont identifiés chacun par un identifiant unique sous la forme "Aid = RSN", "Pid = RSTF". En tant que solution de stockage de la provenance, ils utilisent MySQL Cluster avec un schéma relationnel défini comme suit :

- Artefact (Aid, V, C) indique que l'artefact Aid a une valeur V et la somme de contrôle de la valeur est C.
- Compress (C, M) indique qu'une somme de contrôle C a une valeur compressée M.
- Acteur (E, T, R) indique qu'un acteur nommé E a un numéro T pour l'exécution du workflow R.
- Dépendance (Aid, Pid, D) indique que l'artefact Aid a été lu ou écrit, spécifié par D, par le processus Pid.

4.3. Interrogation et visualisation des métadonnées de provenance

Ils avaient développé une API Query pour interroger les informations de provenance stockées dans le modèle relationnel du cluster MySQL à l'aide de requêtes SQL, et visualiser les résultats de ces requêtes sous forme de graphe.

4.4. Contribution

La principale contribution de cette approche est d'intégrer la provenance au niveau de "Kepler", permettant ainsi la capture de la provenance lors de l'exécution de workflows basés sur MapReduce dans l'environnement Hadoop, ce qui n'est pas pris en charge dans les travaux antérieurs.

4.5. Limites

L'intégration de la gestion de provenance dans Kepler a présenté quelques désavantages :

- L'utilisation de la terminologie W3C OPM pour décrire les entités et les relations capturées au lieu de W3C PROV, qui fournit des types de relation beaucoup plus riches que OPM.
- Le stockage de la provenance étant fait dans une base de données relationnelle "MySQL".
- Pas de prise en charge des exigences non fonctionnelles telles que la sécurité.
- L'utilisation la méthode eager pour la capture de provenance, ce qui a l'inconvénient de payer à l'avance les frais généraux de capture de provenance pour toutes les transformations.

5. Approche de M. Interlandi et al [24, 25]. (2018)

M.Interlandi et al proposent Titian, une bibliothèque améliorant l'orthographe RDD avec des capacités de provenance de données (suivi des données via des transformations) dans le système Data-Intensive Scalable Computing (DISC) Apache Spark pour identifier rapidement les données d'entrée à l'origine d'un bug potentiel ou d'un résultat aberrant via une simple interface de programmation d'application Lineage RDD.

5.1. Capture des métadonnées de provenance

Cette bibliothèque utilise des agents pour la capture et le stockage de lignage de données tels que :

- Les agents d'entrée génèrent et attachent un identifiant unique à chaque enregistrement d'entrée. Ces agents sont l'agent 'Hadoop Lineage RDD', qui attribue un identifiant indiquant la partition HDFS et la position d'enregistrement, ainsi que l'agent 'ParallelLineageRDD', qui attribue des identifiants aux enregistrements en fonction de leur emplacement dans un objet de collection Java.
- Les agents d'agrégation génèrent des identifiants uniques pour chaque enregistrement de sortie et associent un enregistrement de sortie à tous les enregistrements d'entrée dans l'opération d'agrégation. Ces agents sont 'ReducerLineageRDD', 'JoinLineageRDD' et 'CombinerLineageRDD', utilisés pour exécuter les transformations de réduction, de regroupement et de jointure.
- Les agents d'état attachent un identifiant unique à chaque enregistrement de sortie d'une étape et l'associent à l'identifiant d'enregistrement d'entrée pertinent. Cet agent est 'StageLineageRDD', utilisé lorsqu'un combinateur n'est pas présent.

Chaque agent a une méthode de calcul 'compute (split : Partition, context : TaskContext) : Iterator[T]' qui retourne un itérateur sur les enregistrements de sortie résultants. Une fois implémentée, cette méthode permettra la capture du lignage de données.

5.2. Stockage des métadonnées de provenance

Les agents sont chargés d'associer les enregistrements de sortie d'une opération (c'est-à-dire, étape, combineur, jointure) avec les entrées correspondantes. Ces associations sont stockées dans une table BlockManager (couche de stockage interne de Spark) qui définit deux colonnes contenant l'ID d'entrée et l'ID de sortie. Enfin, pour l'interrogation interactive des données de lignage stockées en mémoire, ils vident ces données dans un magasin externe "HDFS".

5.3. Interrogation des métadonnées de provenance

Le traçage est implémenté en joignant de manière récursive les tables d'association de lineage stockées dans le BlockManager, et pour obtenir le lineage dans les deux sens, il exécute la méthode 'getLineage', donnant comme sortie un RDD Lineage qui possède quatre fonctions (requête provenance) pour le parcourir : 'goBackAll', 'goNextAll', 'goBack', 'goNext'. Tout cela se fait via une interface programmable LineageRDD simple (voir figure 4.5).

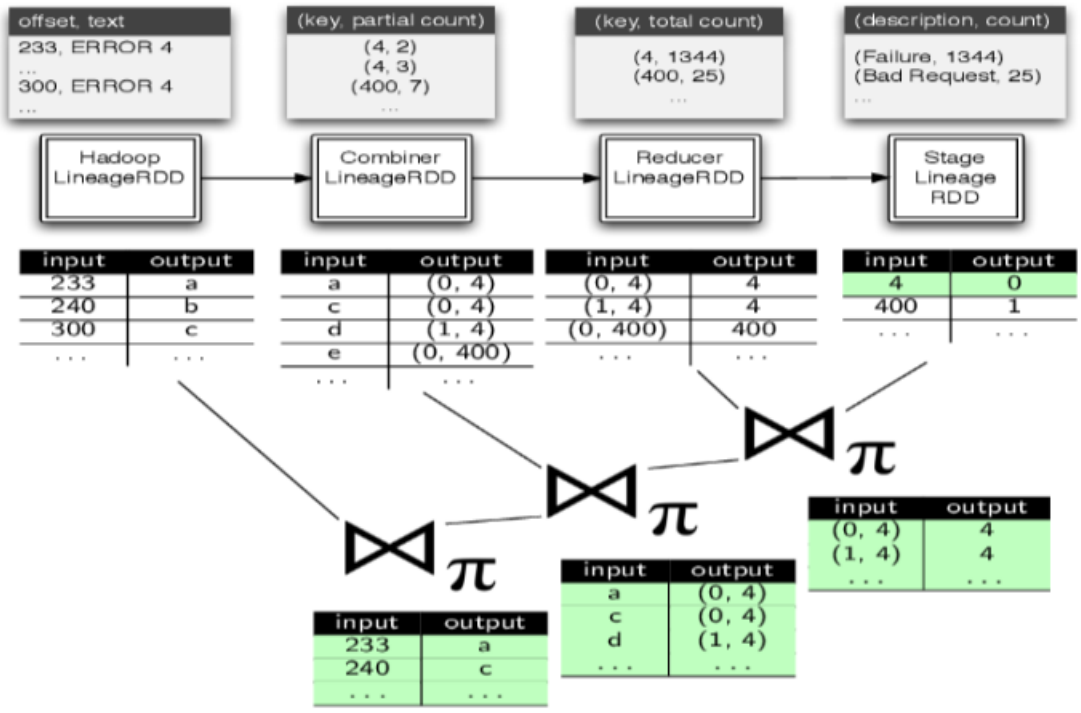


Figure 4.5: Mécanisme de traçage dans Titian.

5.4. Contribution

L'objectif de cette approche est de fournir une prise en charge de la provenance s'intégrant au modèle de programmation DISC, permettant ainsi le débogage de données qui est un défi, en aidant à identifier les données d'entrée ayant causé un crash, une exception ou des résultats erronés. Les précédentes approches telles que NEWT ou RAMP présentent les limitations suivantes :

- Elles utilisent un stockage externe pour enregistrer les informations de provenance capturées, comme HDFS ou une DBMS partagée, ce qui fait qu'elles ne fonctionnent pas bien à grande échelle.
- L'interrogation de la provenance se fait via une interface programmable séparée.
- Elles offrent peu de support pour la visualisation des données intermédiaires ainsi que des traitements effectués sur celles-ci.

Titian améliore l'abstraction RDD (structure de base dans Spark) avec une provenance finement détaillée.

4.5. Limites

On peut citer :

- Les données de provenance capturées sont limitées.
- Ne prend pas en charge les exigences non fonctionnelles telles que la sécurité.

6. Discussion

Le but des quatre approches précédemment étudiées est la collecte, le stockage, l'interrogation ainsi que la visualisation de la provenance. Chacune propose sa propre architecture avec différents outils et modèles et utilise différents datasets.

Elles sont conçues pour des environnements différents. En effet, les deux approches prennent en charge, la capture de la provenance dans un environnement Data Lake, l'approche H. Dibowski (Semantic Data Lake Catalog) et al et I. Suriarachchi et B. Plale (KOMADU) et l'approche de J. Wang et al (KEPLER), est propre à la provenance générée au niveau d'un traitement MapReduce, enfin, l'approche M. Interlandi (TITIAN) et al, propose une solution pour les DISC (Data-Intensive Scalable Computing) en particulier SPARK.

Ces quatre approches diffèrent par leurs mécanismes. Certaines d'entre comme TITIAN et KOMADU utilisent des méthodes de capture paresseuses (lazy), tandis que d'autres comme KEPLER sont plus hâtives (eager), sachant que les méthodes hâtives peuvent engendrer des frais supplémentaires.

Chacune de ces contributions présente des limites. Par exemple, certaines se concentrent uniquement sur les dépendances des données comme H. Dibowski (Semantic Data Lake Catalog) et I. Suriarachchi et B. Plale (KOMADU) ou les dépendances des transformations comme l'approche J. Wang et al (KEPLER) et M. Interlandi et al (TITIAN), ainsi que sur le stockage des données de provenance dans des bases de données relationnelles. C'est là que notre contribution intervient, en proposant un stockage distribué dans une base de données NoSQL. De plus, nous aborderons les dépendances entre les données et les dépendances entre les transformations.

À partir de l'étude des articles sélectionnés, nous comparons les différentes approches selon un ensemble de dimensions présentées dans le tableau 4.1.

		Approche 1	Approche 2	Approche 3	Approche 4
Aspect provenance					
Data access	Querying	✔	✔	✔	✔
	accessin	✔		✔	
tracing	lazy		✔		✔
	eager	✔		✔	
contenu	ligneage	✔	✔	✔	✔
	where			✔	✔
	why	✔	✔	✔	✔
	who	✔	✔		
orientation	data-oriented	✔	✔		
	process-oriented			✔	✔
Aspects du système					
Le modèle de provenance		PROV	PROV	OPM	N/A
availability(open source)		non	oui	oui	oui
Type d'architecture		Centralisée	Distribuée/ centralisée	Distribuée	Distribuée
Stockage de provenance		Stardog	Base de données MySQL	Base de données MySQL cluster	HDFS
Système de traitement des données		DevOps	Apache flume, Apache Hadoop, Apache spark	MapReduce	Hadoop Spark
Modèle NoSQL		N/A	Clé /valeur	Clé/valeur	Clé/valeur
Système de stockage des données		N/A	HDFS	HDFS	HDFS
Implémentation		Oui	Oui	oui	oui
Types de données		Structurées, Semi structurées, Non structurées	Structurées, Semi structurées, Non structurées	Structurées, Semi structurées, Non structurées	Structurées, Semi structurées, Non structurées
Dataset		Données de capture auto mobilier	Données de Twitter	Produit Scientifique	Produit Scientifique
Taille dataset		64.1648 Petabytes	3,23 Go	N/A	500 Mo à 500Go
Environnement		Data Lake	Data Lake	Hadoop/ Cloud/cluster	HADOOP
réplication des données		N/A	3 fois	2 fois	3 fois
Informations sémantiques		Oui	Oui	oui	non
Outils sémantique		Ontologie	XML	N/A	N/A
Interrogation et visualisation de la provenance		Interface API, requête Spark	Interface API Query API	Interface client Kepler Query API	Query
Plateforme d'implémentation		Data lake Catalog web application	komadu	Kepler	librairie Titian
Langages utilisés		SKOS,RDF, OWL, React	Java	Java	Scala
Hosting		Cloud, Docker	On -premise	Cloud	On-premise

Tableau 4. 1: Comparaison des travaux connexes.

7. Conclusion

L'étude de la littérature concernant la provenance nous a permis de comprendre sa définition, les défis auxquels elle est confrontée, ses utilisations ainsi que la façon dont ses données sont gérées à l'aide d'outils informatiques.

L'approche de Komadu est celle qui se rapproche le plus de notre travail, car elle est conçue dans un environnement Data Lake et capture les données relatives à la provenance, telles que l'identité de leurs créateurs, les informations de provenance concernant les traitements effectués et les données utilisées. Le sous-système de provenance prend en charge la provenance générée par Flume, Hadoop et Spark.

Dans ce chapitre, nous avons abordé quelques contributions récentes sur les systèmes de provenance, en examinant leurs avantages et leurs limites. Enfin, nous avons dressé un tableau comparatif de ces outils selon différents critères et avons mené une discussion sur les approches définies.

VI. Proposition d'une nouvelle approche de capture des données de provenance dans un environnement Data Lake.

Ce chapitre présente notre contribution dans le cadre de ce travail consistant au "Data Lineage et Data Provenance dans un environnement DATA LAKE". La présentation de ce chapitre est comme suit : introduction, description de la solution proposée, un algorithme décrivant le fonctionnement de notre système de capture, et la conclusion vers la fin.

---

1. Introduction

Nous avons présenté dans ce qui précède les concepts fondamentaux liés à la provenance des données.

Dans ce chapitre, nous décrivons notre approche pour le Lineage des données dans un environnement Data Lake. Puisque celui-ci regroupe principalement des données de tous types, il est capital de pouvoir tracer la cartographie du cycle de vie de ces données afin d'en tirer meilleur profit. Il existe ainsi deux catégories d'informations de provenance : celles liées aux données et celles liées aux traitements. Le but de notre plateforme est donc de répondre à ces questions concernant la provenance : qui a créé un produit de données spécifique et quand ? qui l'a modifié et quand ? quel processus a créé le produit de données ? quels produits de données ont été dérivés de ce produit de données ? Quelles sont les sorties et les entrées d'un processus ? qui a exécuté un traitement spécifique ? De cette manière, nous pourrons également connaître les processus les plus utilisés, obtenir des informations sur les utilisateurs du système, tracer leur historique d'utilisation et leurs préférences.

Notre système prendra en charge la capture, le stockage ainsi que la visualisation des informations de provenance. La provenance sera générée au fur et à mesure des traitements effectués sur les données brutes et sera stockée dans un SGBD NoSQL orienté graphe (Neo4j), étant donnée sa bonne scalabilité et sa flexibilité, pour finalement être restituée sous forme graphique et textuelle.

2. Architecture du système

L'architecture du système proposée (voir figure 6.1) est une solution de capture des détails relatifs à la création et à la modification des sources de données du Data Lake, ainsi que le stockage et la visualisation des métadonnées de provenance capturée. Elle est composée de :

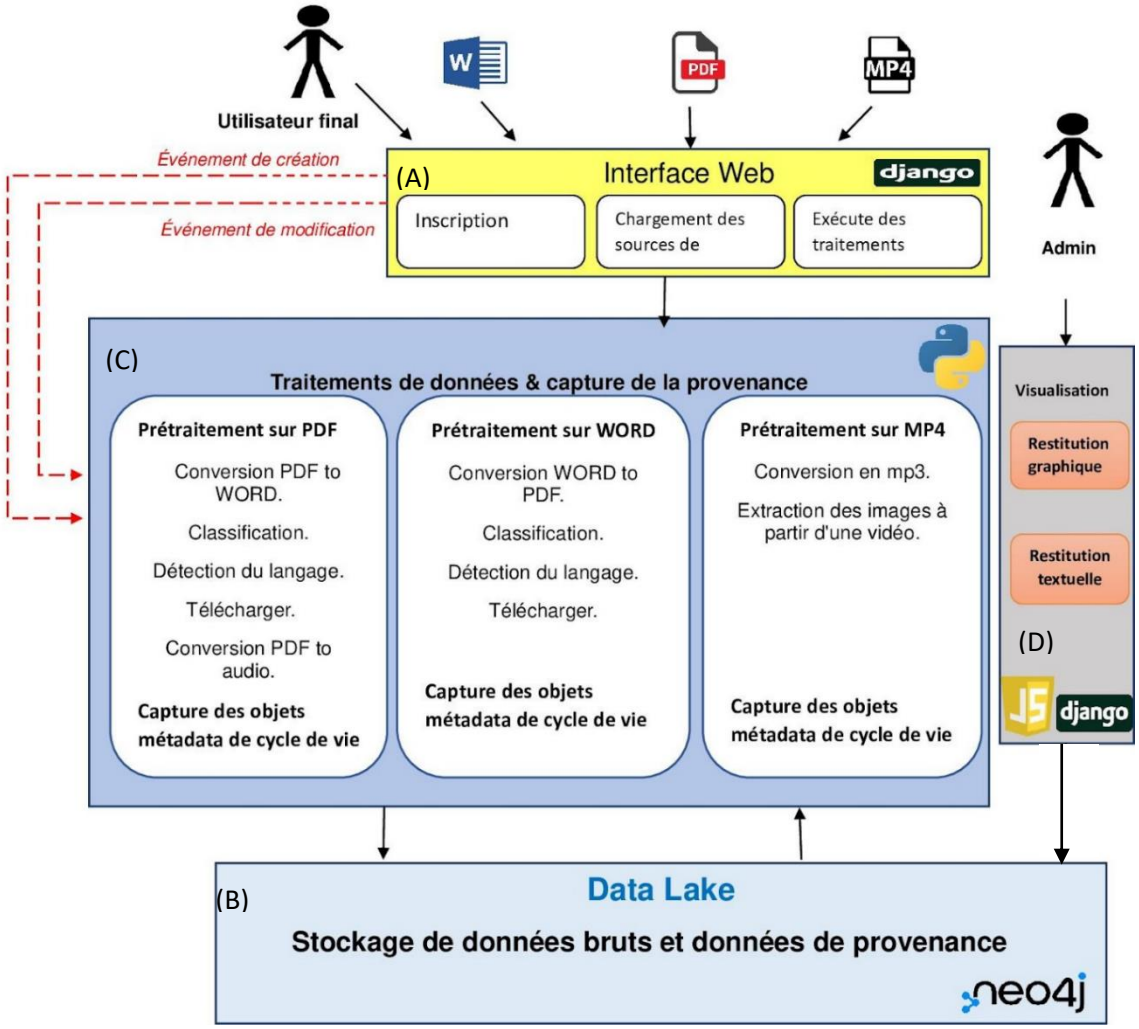


Figure 6. 1: Architecture proposée pour la gestion de la Data Provenance et Data Lineage.

A. Interface WEB

Cette interface offre la possibilité d'ajouter des données brutes au data Lake manuellement et d'exécuter des traitements sur elles. L'utilisateur doit s'authentifier pour pouvoir y accéder et effectuer des traitements permettant à notre système de provenance de capturer les informations concernant les agents ayant créé et modifié les données du data Lake.

B. Environnement de stockage

Dans notre système, le Data Lake représente notre environnement de stockage de données brutes ainsi que les données résultantes des prétraitements, ainsi que les informations de provenance. Il a la capacité d'ingérer tous types de données. Nous utilisons le SGBD Neo4j qui permet un stockage distribué des données, ainsi que le stockage des informations de provenance et des métadonnées relatives à la donnée elle-même.

C. Système de traitements de données et capture de la provenance

Notre système de capture de provenance capture les événements de création ainsi que ceux de la modification des données stockées au sein du Data Lake selon les algorithmes qui suivent, tels que, à chaque création ou modification, des nœuds et relations traduisant les informations de provenance sont ajoutés (données, processus, agent, métadonnées supplémentaires), formant ainsi un graphe de provenance en cours de génération stocké dans le SGBD Neo4j, qui sera détaillé davantage dans la section 3.

D. Interface de visualisation

Cette interface est consacrée à la visualisation des métadonnées de provenance capturées et prises en charge dans notre approche.

3. Description du système

Dans ce qui suit, la description du fonctionnement de notre système par le biais d'un diagramme de classe, d'utilisation et des algorithmes :

3.2. Description textuelle

À l'arrivée d'un événement de création, notre système stocke la donnée brute dans le data Lake 'Neo4j' avec un ID que nous générons et capture à travers un code python les informations de l'agent qui a créé la donnée, telles que : @ IP, informations d'usage, leurs informations de localisation, le processus de création et leurs métadonnées, ainsi que les métadonnées de la donnée elle-même, telles que la taille, le format, le type (voir l'algorithme 1). Nous utilisons des bibliothèques de l'intelligence artificielle, telles que 'NLTK', pour classifier les fichiers selon des catégories, ainsi que 'LANGDETECT' pour détecter la langue, et les stockons dans la base de données 'Neo4j'. Chaque nœud de donnée contient des métadonnées relatives à la donnée elle-même, et nous ajoutons l'ID que nous générons dans le nœud Data pour nous permettre d'y accéder. Nous établissons les relations "executed_by" entre l'agent et le processus pour définir quel agent exécute quel processus, "generated_by" entre la donnée et le processus indiquant l'activité ayant généré la donnée sélectionnée, et "created_by" qui indique l'agent responsable de la création de la donnée. Nous utilisons également le champ "type" pour indiquer le type de données (structurées, non-structurées, semi-structurées), et la relation "additional_metadata" qui relie les données au nœud contenant des métadonnées spécifiques selon le format de données. Enfin, nous utilisons la relation "classified" entre les données et leur classe (littéraire, scientifique, culturelle, journalistique, historique, politique). Nous avons donc capturé la provenance liée aux données et celle liée aux traitements, comme indiqué dans la figure 6.2.

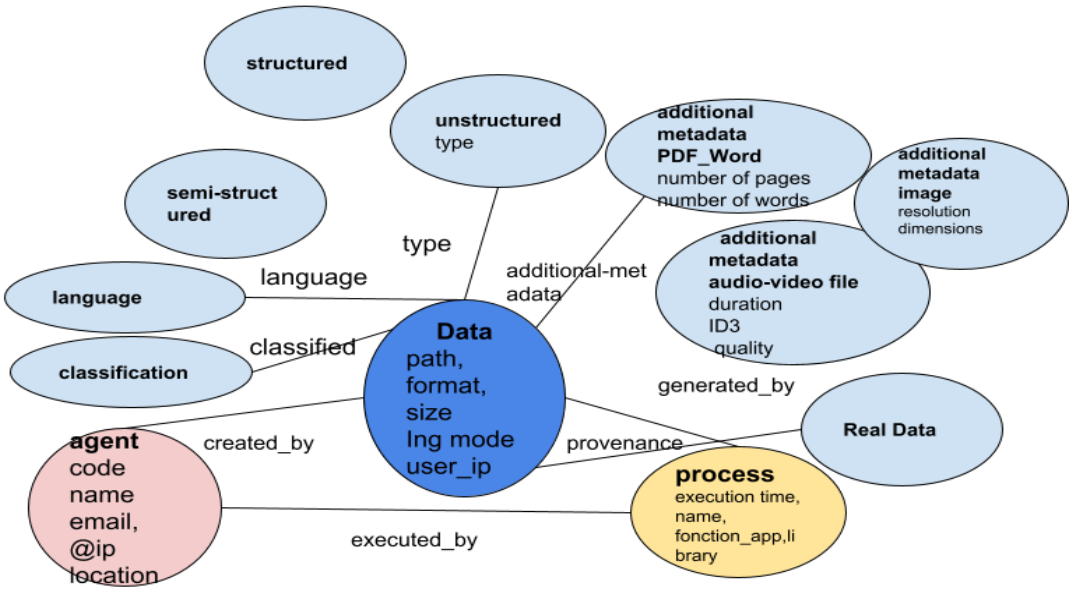


Figure 6. 2: Graphe de provenance.

Dans le cas de l'arrivée d'un événement de modification (prétraitement) d'une donnée : La donnée résultante est stockée dans le Data Lake sous format brut avec un ID, est reliée avec la donnée d'entrée grâce à la relation derived_from. L'agent ayant effectué la modification est relié avec la donnée d'entrée grâce à la relation edit_by et avec la donnée résultante (de sortie) grâce à created_by, comme nous le montrons dans la figure 6.3. On relie la donnée résultante avec les nœuds ayant leurs métadonnées supplémentaires et leur type et leur classification selon le contenu si la donnée est un Word ou un PDF par les relations additional_metadata, type et classified(voir l'algorithme 2).

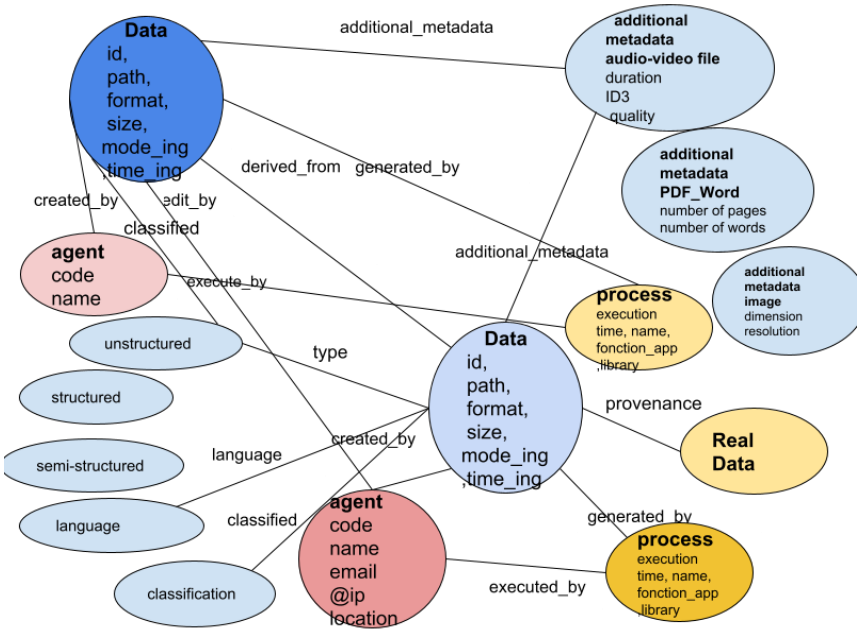


Figure 6. 3: Graphe de provenance.

3.4. Diagramme de cas d'utilisation

Notre système comprend deux types d'acteurs : l'ingénieur du Data Lake qui s'intéresse au cycle de vie des données et l'utilisateur du système qui visualise, accède et effectue des prétraitements sur les données du lac selon ses besoins. La figure 6.4 représente le diagramme de cas d'utilisation décrivant la responsabilité de chaque acteur dans notre système et ses interactions avec.

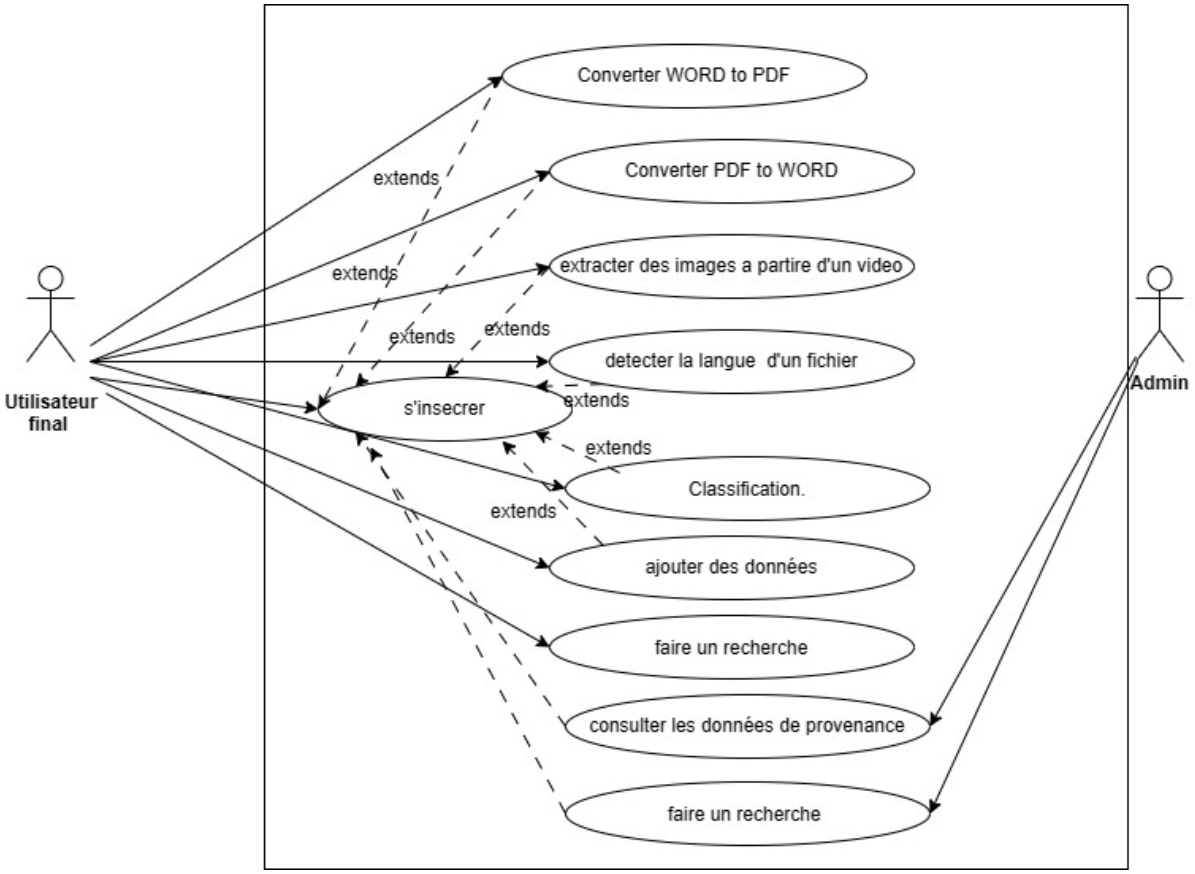


Figure 6. 4: Diagramme de cas d'utilisation.

3.4. Diagramme de classes métadonnées provenance

La figure 6.5 représente le diagramme de classe global de notre système qui décrit ses fonctionnalités de manière détaillée ainsi que les entités présentes dans notre Data Lake, ainsi que les relations entre elles.

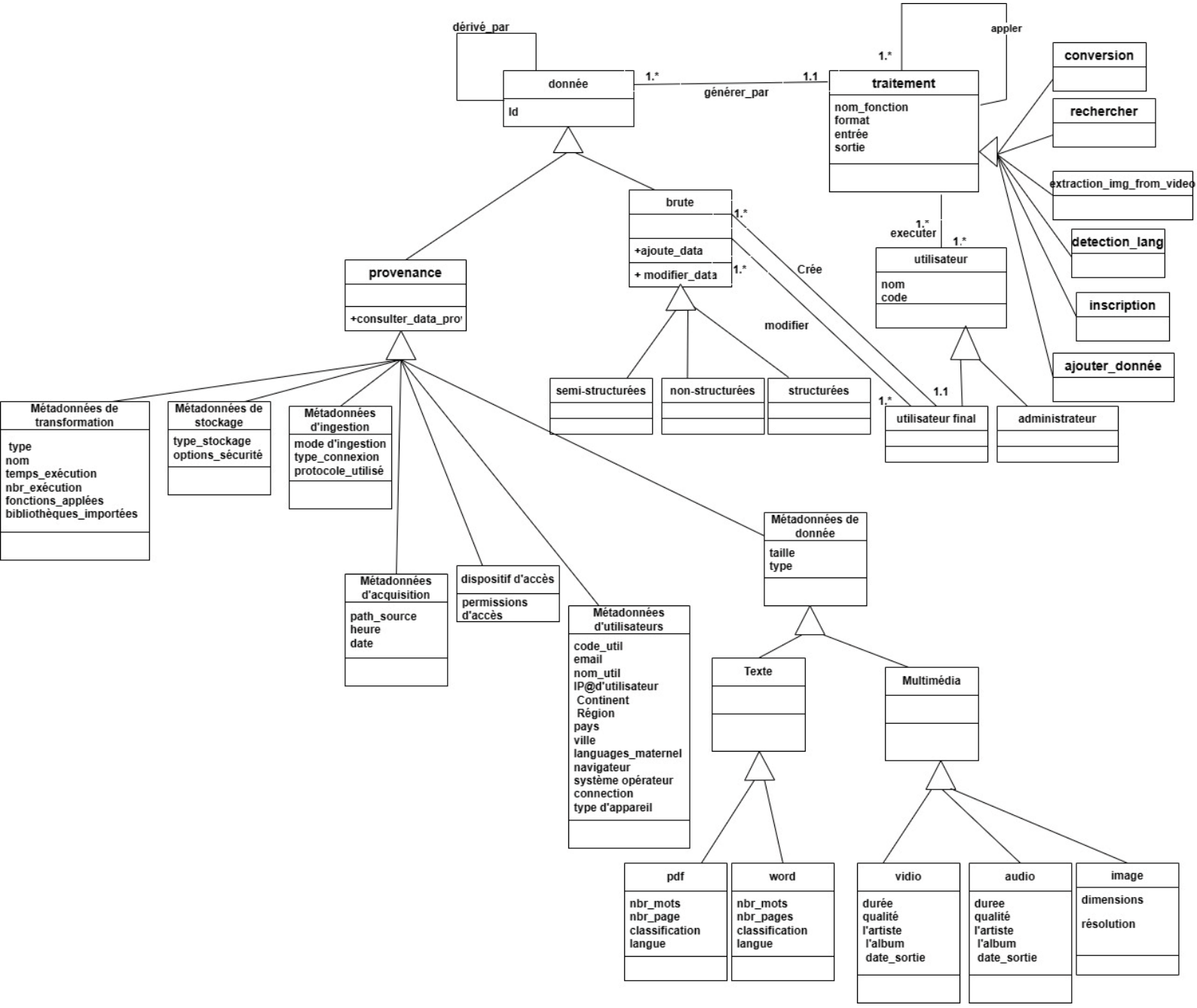


Figure 6. 5: Diagramme de classes métadonnées provenance.

3.1.1 Description du diagramme de classe

a) Classes

Classe	Description	Attribut	Description
Données	Classe regroupant tous les type de données présentes dans le data Lake	Id	Identifiant

Brutes	Classe des données d'origine et données résultantes		
Structurées	Type de données brute		
Semi-structurées	Type de données brute		
Non-structurées	Type de données brute		
Provenance	Classe incluant toutes les données de provenance capturées		
Métadonnées sur les transformations	Classe incluant des métadonnées pour décrire les traitements exécutés dans le système	Nom	Nom des traitements appliqués sur les données
		Type	Les étapes du traitement
		Temps_execution	Date et heure d'exécution
		Nbr_exécution	Nombre de fois d'exécution du traitement
		Bibliothèques_importées	Bibliothèques utilisées pour les traitements
		Fonctions_appelées	Fonctions appelées lors du traitement
Métadonnées sur le stockage	Classe fournissant des détails sur les caractéristiques du stockage utilisé	Type_stockage	{Centralisé, distribué}
		Options_sécurité	Niveau de sécurité {bas, moyen, élevé}
Métadonnées sur l'ingestion	Inclut les métadonnées décrivant l'intégration des données dans le data Lake	Mode d'ingestion	{manuel, automatique}
		Type_connexion	Ethernet, Wi-Fi, Bluetooth, VPN
		Protocole_utilisé	Protocole utilisé pour transférer les données du point de collecte au système de stockage {FTP,HTTP,...}
Métadonnées sur l'acquisition	Cette classe inclut les métadonnées capturées sur la donnée lors de l'ingestion	Path_source	Source de la donnée
		Heure	Heure d'ingestion
		Date	Date d'ingestion
Dispositif d'accès	Classe décrivant les métadonnées	Permission d'accès	{Privé, publique,

	sur les privilèges d'accès		…}
Métadonnées sur l'utilisateur	Classe de métadonnées servant aux profilage des utilisateurs	Code_utilisateur	Identifiant de l'utilisateur
		Email	Adresse mail de l'utilisateur
		Nom_util	Nom d'utilisateur
		IP@utilisateur	Adresse IP de l'utilisateur
		Continent	Information de Localisation
		Région	
		Pays	
		Ville	
		Language_maternel	{Français, arabe, anglais, …}
		Navigateur	{Chrome, Mozilla, …}
		Système opérateur	{Windows, linux,}
		Connexion	Ethernet, Wi-Fi, Bluetooth, VPN
		Type d'appareil	{Pc, téléphone,}
Métadonnées sur la donnée	Classe incluant une description générale des données	Taille	Taille de la donnée en octets
		Type	{Structurée, semi structurée, non structurée}
Texte	Classe dérivée incluant les fichier PDF et Word		
Word	Classe dérivée de la classe texte incluant des métadonnées spécifiques aux fichier Word	Nbr_pages	Nombre de pages de fichier Word
		Nbr_mots	Nombre de mots de fichier Word
		Classification	{Historique, scientifique, politique, journalistique…}
		Langue	{Français, anglais, arabe….}
PDF	Classe dérivée de la classe texte incluant des métadonnées spécifiques aux fichier PDF	Nbr_pages	Nombre de pages de fichier PDF
		Nbr_mots	Nombre de mots de fichier PDF
		Classification	{Historique, scientifique, politique, journalistique…}

		Langue	{Français, anglais, arabe…}
Multimédia	Classe dérivée pour décrire les données multimédia		
Audio	Classe dérivée de multimédia pour décrire les métadonnées spécifiques aux fichiers MP3	Durée Qualité L`artiste L'album Date_sortir	ID3 : propriétés communes à tous les fichier multimédia
Vidéo	Classe dérivée de multimédia pour décrire les métadonnées spécifiques aux fichiers MP4		
Image	Classe dérivée de multimédia pour les métadonnées spécifiques aux fichiers PNG	Résolution	Nombre de pixel contenus dans un pouce
		Dimensions	Longueur et largeur
Traitements	Classe incluant tous les traitements possibles dans notre système	Nom_fonction	Nom du traitement exécuté
		Format	Format des données traitées {PDF, WORD, MP4}
		Entrée	Donnée en entrée
		Sortie	Donnée en sortie
Conversion **Detection-lang** **Ajouter-fichier** **Inscription** **Extraction_img_from_video**	Classes dérivées de la classe traitements		
Utilisateurs	Classe incluant les utilisateurs du système	Nom	Nom de l'utilisateur
		Code	Identifiant
Administrateur	Classe dérivée de la classe utilisateurs (qui visualisent la provenance)		
Utilisateur final	Classe dérivée de la classe utilisateurs (qui exécutent les traitement)		

Tableau 6. 1: Description des classes.

b) Méthodes

Classes	**Méthodes**	**Description**
Brute	Ajouter_data	Fonction exécutée par l'utilisateur final sur les

		données brute permettant d'intégrer des fichiers dans le data Lake
Brute	Modifier_data	Fonction exécutée par l'utilisateur final sur les données brute permettant de modifier des fichiers dans le data Lake
Provenance	Consulter_data_prov	Fonction exécutée par l'administrateur pour consulter les données de provenance capturées

Tableau 6. 2: Description des méthodes.

c) Associations

Associations	**Description**
Derivée_par	Relation entre la donnée entrante et ses données résultantes
Générée_par	Relation entre données du Data Lake et les traitements responsables de leurs création
Appeler	Relation entre traitements puisqu'un traitement peut nécessiter l'exécution d'un autre traitement
Exécuter	Relation entre utilisateurs et traitements, un utilisateur pouvant exécuter plusieurs traitements et un traitement exécuté par plusieurs utilisateurs
Créer	Relation entre données brutes et utilisateurs finaux, un utilisateur pouvant créer plusieurs données et une donnée crée par un seul utilisateur final
Modifier	Relation entre données brutes et utilisateurs finaux, un utilisateur final peut modifier plusieurs données et celles-ci peuvent être modifiées par plusieurs utilisateurs finaux

Tableau 6. 3: Description des associations.

3.4. Algorithmes

Pour une description plus pointue de la démarche suivie dans notre approche pour capturer les métadonnées de provenance, on présente les deux algorithmes suivants :

Algorithme 1 : évènement de création

```
Entrée :
Data : Fichier,
Variable :
Fonction Capture_Traitement (Data) : Liste,
    Variable :
    List1 : Liste,
    Début
        List1 <-- (nom_fonction, temp_execution, nbr_execution, fonctions_applées, bibliothèque),
    Fin,
Retourner (Liste1),
Fonction Capture_utilisateur (Data : fichier) : Liste
  Variable :
  List2 : Liste,
  Début
    List2 <-- (nom_utilisateur , code, email, location, région, pays, ville, système exploitation,
    Navigateur, type_appareil, @ip, access_permissions),
  Fin,
Retourner (Liste2),
Fonction Capture_donnée (Data : fichier) : Liste
  Variable :
  List3 : Liste,
  Début
    List3 <-- (path, taille, format, temps_ingestion, permissions d'accès),
  Fin
 Retourner (Liste3),
Fonction Capture_metadonnéeSuplimentaire (Data : fichier) : Liste,
  Variable :
  List4 : Liste,
  Début
    Si (format =`PDF`) ou (format =WORD`) alors
      List4 <-- (nbr_word, nbr_pages, language, classification),
    Sinon
      Si (format =`mp4`) ou (format =mp3`) alors
        List4 <-- (durée, qualité, nom_artiste, date_sortie)
      Finsi,
      Si (format =`jpg`) alors
        List4 <-- (dimensions, resolution)
      Finsi,
    Finsi,

  Fin
Retourner (Liste4),

Début
  provenance_utilisateur ← Capture_utilisateur(Data),
  provenance_traitement ← Capture_Traitement(Data),
  provenance_donnée ← Capture_donnée (Data),
  Additional_metadata ← Capture_metadonnéeSuplimentaire(Data),
  N1 ← créer_Noeud (Data provenance_donnée),
  N2 ← créer_Noeud(provenance_traitement),
  N3 ← créer_Noeud(provenance_utilisateur),
  N4 ← créer_Noeud(type),
  N5 ← créer_Noeud (additional metadata),
  N6 ← créer_Noeud (Data),
  N7 ← créer_Noeud(classification),
  N8 ← créer_Noeud(langue),
  Relation (N1, N2, created_by),
  Relation (N1, N3, generated_by),
```

```
    Relation (N2, N3, executed_by),
    Relation (N1, N4, type),
    Relation (N1, N5, additional_metadata),
    Relation (N1, N6, provenance),
    Relation (N1, N7, classified),
    Relation (N1, N8, langue),

Fin,
```

Algorithme 6. 1: évènement de création.

Algorithme 2 : évènement de modification

```
Entrée :
Data : Fichier,
Variable :
Fonction Capture_Traitement (Data) : Liste,
    Variable :
    List1 : Liste,
    Début
        List1 <--(nom_fonction, temp_execution, nbr_execution, fonctions_applées, bibliothèque),
    Fin,
Retourner (Liste1),
Fonction Capture_utilisateur (Data : fichier) : Liste
  Variable :
  List2 : Liste,
  Début
    List2 <-- (nom_utilisateur, code, email, location, région, pays, ville, système exploitation,
    Navigateur, type_appareil, @ip, access_permissions),
  Fin,
Retourner (Liste2),

Fonction Capture_donnée (Data : fichier) : Liste
  Variable :
  List3 : Liste,
  Début
    List3 <-- (path, taille, format, temps_ingestion, permissions d'accès,),

  Fin,
Retourner (Liste3),
Fonction Capture_metadonnéeSuplimentaire (Data : fichier) : Liste
  Variable :
  List4 : Liste,
  Début
    Si (format =`PDF`) ou (format =WORD`) alors
    List3 <-- (nbr_word, nbr_pages, language, classification),
    Sinon
    Si (format =`mp4`) ou (format =mp3`) alors
      List3 <-- (durée, qualité, nom_artiste, date_sortie)
    Si (format =`jpg`) alors
      List3 <-- (dimensions, resolution)
  Fin,
Retourner (Liste3),
 Début
 D2 ← Traitement (Data)
provenance_utilisateur ← Capture_utilisateur(D2)
provenance_traitement ← Capture_Traitement(D2)
```

```
45 provenance_donnée ← Capture_donnée(D2)
46 Additional_metadata ← Capture_metadonnéeSuplimentaire(D2)
47 N1 ← créer_Noeud(provenance_donnée)
48 N2 ← créer_Noeud(provenance_agent)
49 N3 ← créer_Noeud(provenance_traitement)
50 N4 ← créer_Noeud (id = id (Data))
51 N5 ← créer_Noeud (additional metadata)
52 N6 ← créer_Noeud(type)
53 N7 ← créer_Noeud (real data)
54 N8 ← créer_Noeud(classification)
55 N9 ← créer_Noeud(langue)
56 Relation (N1, N4, derived_from)
57 Relation (N1, N2, created_by)
58 Relation (N1, N3, generated_by)
59 Relation (N2, N3, executed_by)
60 Relation (N1, N6, type)
61 Relation (N4, N2, edited_by)
62 Relation (N1, N5, additional_metadata)
63 Relation (N1, N6, additional_metadata)
64 Relation (N7, N1, provenance)
65 Relation (N1, N8, classified)
66 Relation (N1, N9, langue)
67 Fin
```

Algorithme 6. 2: évènement de modification.

4. Proposition d'une algèbre de manipulation d'un graphe de Data Provenance

Dans le but de permettre une restitution plus précise et selon plusieurs dimensions, nous proposons une algèbre de visualisation qui sera prochainement traduite en un ensemble de filtres.

4.1. Dimension utilisateurs : consiste à restreindre la vue par rapport à un utilisateur sélectionné. Cette restriction se traduit par l'expression algébrique suivante :

$$\sigma_{\text{utilisateur}}(Gg) = GU$$

Utilisateur : est un élément de l'ensemble des utilisateurs du système

GU : est le résultat de la restitution relative à l'utilisateur sélectionné

4.2. Dimension données : De même que l'opération précédente, elle se traduit par une restriction, mais cette fois-ci au niveau des données. Cela signifie qu'on aura une vue sur les métadonnées en fonction des données. On peut la traduire comme suit :

$$\sigma_{\text{donnée}=\alpha}(G_g) = G_{D\alpha}$$

Donnée : appartient à l'ensemble des données du Data Lake.

GD : est le résultat de la restitution relative à la donnée spécifiée.

4.3. Dimension traitements : restriction de la vue selon un type spécifique de traitement. Cette restriction se traduit par l'expression algébrique suivante :

$$\sigma_{\text{traitement née=P}} = (G_g) = G_{Tp}$$

P : l'un des traitements effectués par l'utilisateur depuis son inscription

GTp : est le résultat de la visualisation relative au traitement sélectionné

Gg : est le graphe global.

4.4. Dimension format : pour visualiser les métadonnées capturées selon le format de la Donnée :

$$\sigma_{\text{format=f}}(\pi_{nom,format}(\text{donnée})) = G_{U'}$$

f ∈ {PDF, WORD, MP4}

GU' : est le résultat de la visualisation par rapport au format voulu.

4.5. Dimension année/mois : pour visualiser les métadonnées capturées selon le mois et l'année d'ajout de la donnée :

$$\sigma_{\text{année=A}}(\pi_{nom,date}(\text{donnée})) = G_{U'}$$

$$\sigma_{\text{mois=m}}(\pi_{nom,date}(\text{donnée})) = G_{U'}$$

4.6. Dimension lieu : pour visualiser les métadonnées capturées selon le lieu de l'utilisateur :

$$\sigma_{\text{région=r}}(\pi_{nom,date}(\text{donnée})) = G_{U'}$$

r ∈ {Nord, Est, Ouest, Sud}

6. Conclusion

Dans ce chapitre, nous avons détaillé notre approche en expliquant l'architecture de notre plateforme, à savoir notre système de gestion de la provenance dans un environnement Data Lake, conçu pour gérer le cycle de vie des informations de Lineage, notamment la capture, le stockage et la visualisation. Nous avons présenté le fonctionnement de notre système en introduisant une description par des algorithmes ainsi qu'une description textuelle. Dans le chapitre suivant, nous présenterons les détails de mise en œuvre de notre système.

VII. Implémentation

Ce chapitre regroupe l'ensemble des logiciels et matériels utilisées dans l'implémentation de notre prototype ainsi que la description de ses fonctionnalités.

1. Introduction

Il existe à ce jour bon nombre de langages de programmation permettant l'implémentation de projets informatiques. Le choix des outils matériels et logiciel se fait selon le type du projet à réaliser, les besoins fonctionnels ainsi que le coût et la maîtrise de ces outils.

2.Environnement de développement

2.1. Matériel utilisé

Nous avons utilisé un ordinateur avec les caractéristiques suivante :

Unité	Caractéristique
Processeur	Intel(R) Core (TM) i7-8550U@ 1.80GHz 2.00 GHz
Mémoire	8.00 Go
Type de système d'exploitation	64-bit operating system, x64-based processor
Edition	Windows 10
Version	21H1

Tableau 7. 1: Caractéristiques du matériel utilisé.

2.2. Plateformes logicielles et langages de programmation

2.2.1. Django

Notre application a été développée avec Django, un Framework open source écrit en Python dans l'environnement de développement "Visual Studio Code". Il est conçu pour la création d'applications web et repose sur l'architecture MVC (Modèle, Vue, Contrôleur), tel que :

- **Le modèle** : Le modèle définit la structure des données de l'application, telles que les tables de base de données et les relations entre les tables.
- **La vue** : C'est la partie logique de l'application, elle se traduit en un ensemble de fonctions qui traitent une requête HTTP et renvoient une réponse HTTP. Elle récupère les données à partir du modèle et effectue des traitements selon les paramètres présents dans la requête HTTP.

- **Le contrôleur** : Est géré par le Framework lui-même et est appelé le routeur (URL dispatcher). Le routeur d'URL est responsable de mapper les URL aux vues correspondantes pour générer une réponse HTTP.
- Le développeur peut définir les informations sur les bases de données et autres paramètres de l'application dans le fichier "settings.py".

Son avantage est qu'il fournit des composants prêts à l'emploi pour faciliter la création d'applications web robustes et sécurisées.

Pour l'utiliser, il faut : Installer Python ainsi que Django avec la commande "pip install django".

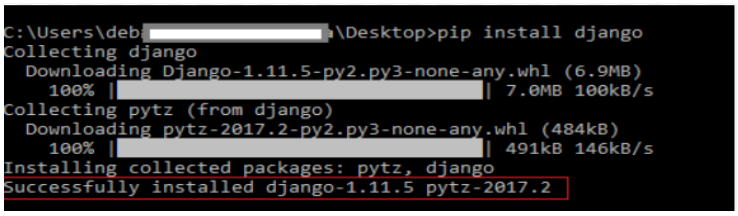


Figure 7. 1: Installation de Django.

2.2.2. Neo4j

Neo4j est un SGBD non relationnel orienté graphe. Nous l'avons choisi pour y stocker nos données brutes, ainsi que les données de provenance et autres métadonnées capturées, puisqu'il est principalement conçu pour analyser les interconnexions entre les données et permettre la visualisation de nos données sous format graphique. Dans Neo4j, les données sont stockées sous forme de nœuds et les relations entre elles sont représentées par des arcs (voir figure 7.2), ce qui justifie notre choix de ce SGBD.

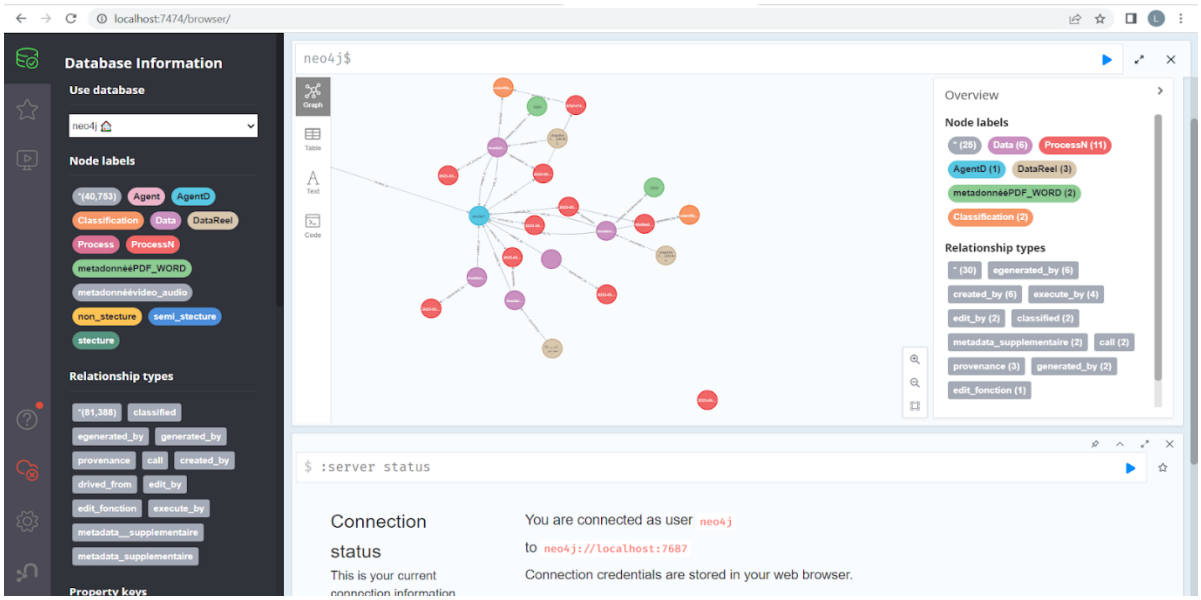


Figure 7. 2: Interface de Neo4j.

2.2.3. HTML/CSS

HTML (HyperText Markup Language) est un langage de balisage utilisé pour créer des pages web, tandis que le CSS (Cascading Style Sheets) est un langage de feuille de style utilisé pour décrire leur apparence et leur mise en page, car il permet de contrôler les couleurs, les polices, etc. Nous l'avons utilisé pour construire notre interface.

2.2.4. JAVASCRIPT

Langage de programmation utilisé pour la création d'applications web, il est également utilisé pour manipuler du contenu HTML/CSS en vue d'obtenir des pages web interactives. Nous l'avons utilisé pour l'interface utilisateur et celle de visualisation avec la bibliothèque popoto.js pour visualiser les résultats des graphes à partir de notre application sans recourir à l'interface Neo4j.

2.2.5. BOOTSTRAP

Cet outil n'est autre qu'un Framework basé sur les langages HTML, CSS et JavaScript. Sur ce, il fournit aux développeurs des composants préconçus afin de simplifier la création d'interfaces plus agréables.

2.2.6. Adobe Photoshop

Photoshop est un logiciel de retouche d'image développé par Adobe. Nous l'avons utilisé pour concevoir le logo de notre plateforme (voir figure 7.3).

Figure 7. 3: Logo de notre plateforme.

3. Description de l'application

Notre application "Data Provenance" est un système de veille sur les données présentes au sein du Data Lake. Elle fournit deux interfaces : une pour l'utilisateur lui permettant d'ajouter des fichiers de type PDF, Word, MP4 dans le Data Lake et d'effectuer des traitements, et l'autre conçue pour l'administrateur afin de visualiser les métadonnées de chaque donnée du Data Lake.

3.1. Côté utilisateur (End-user)

Espace d'authentification

Pour accéder à la page d'accueil, l'utilisateur doit créer un compte en entrant un nom d'utilisateur, une adresse e-mail, un code et un mot de passe. S'il existe déjà dans la base de données, il lui suffit alors de s'authentifier avec son nom d'utilisateur et son mot de passe (figure 7.4).

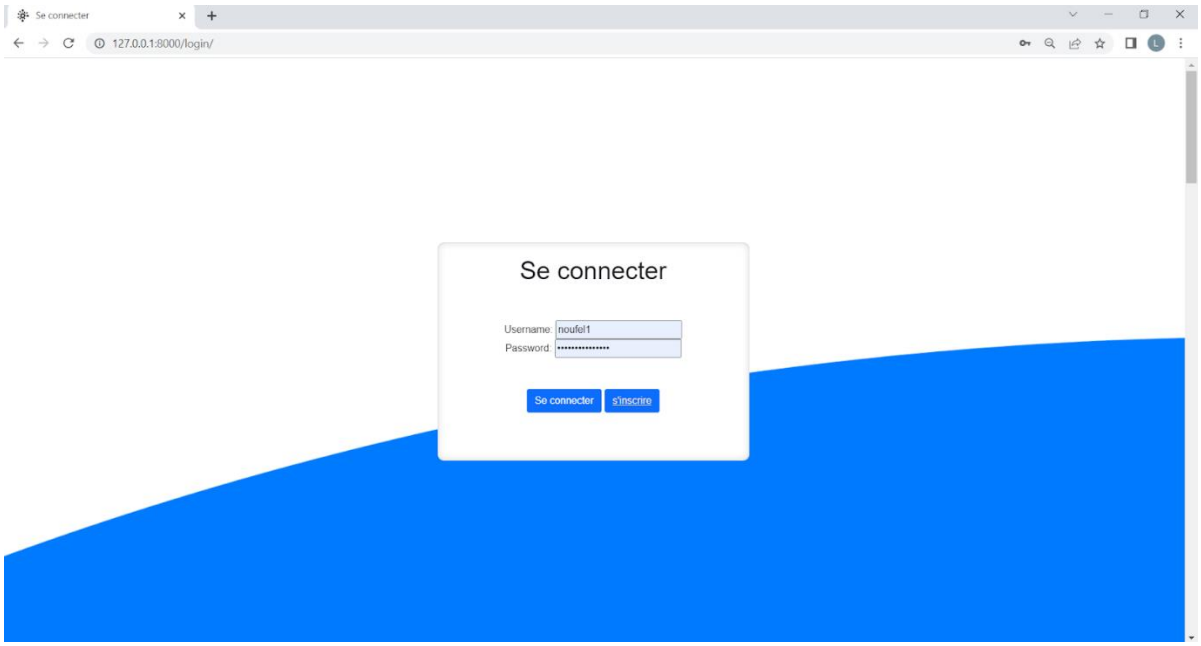


Figure 7. 4: Capture d'écran de l'espace d'authentification.

Page d'accueil

Cette page s'affiche directement après l'authentification. L'utilisateur peut choisir d'ajouter des fichiers dans le Data Lake ou exécuter les traitements proposés dans l'application. Les fichiers sont classés par leurs types et possèdent, pour chaque type, un ensemble de traitements spécifiques, comme le montre la figure 7.5.

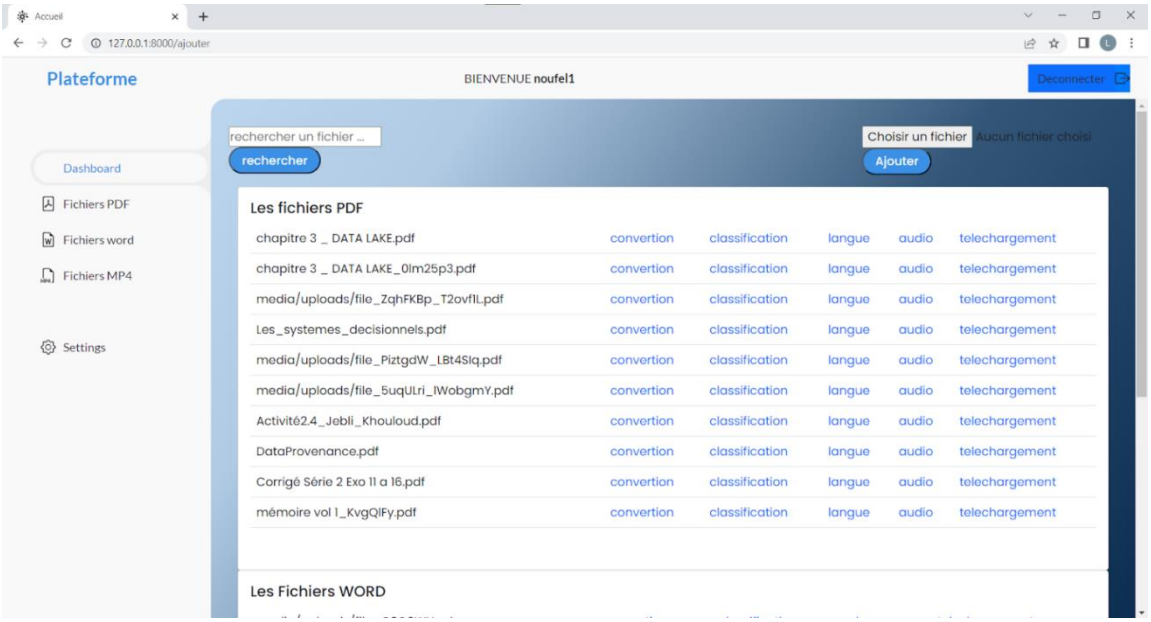


Figure 7. 5: Capture d'écran de page d'accueil.

3.2. Côté administrateur

Page d'accueil de l'administrateur

Cette page est conçue pour l'administrateur du Data Lake. Elle lui permet de visualiser les informations des métadonnées selon l'utilisateur, la donnée, le traitement et autres filtres en cliquant sur les options à gauche. On peut également accéder aux métadonnées en indiquant le nom d'une donnée, un traitement ou un utilisateur dans la barre de recherche au milieu de l'écran, comme capturé dans la figure. 7.6

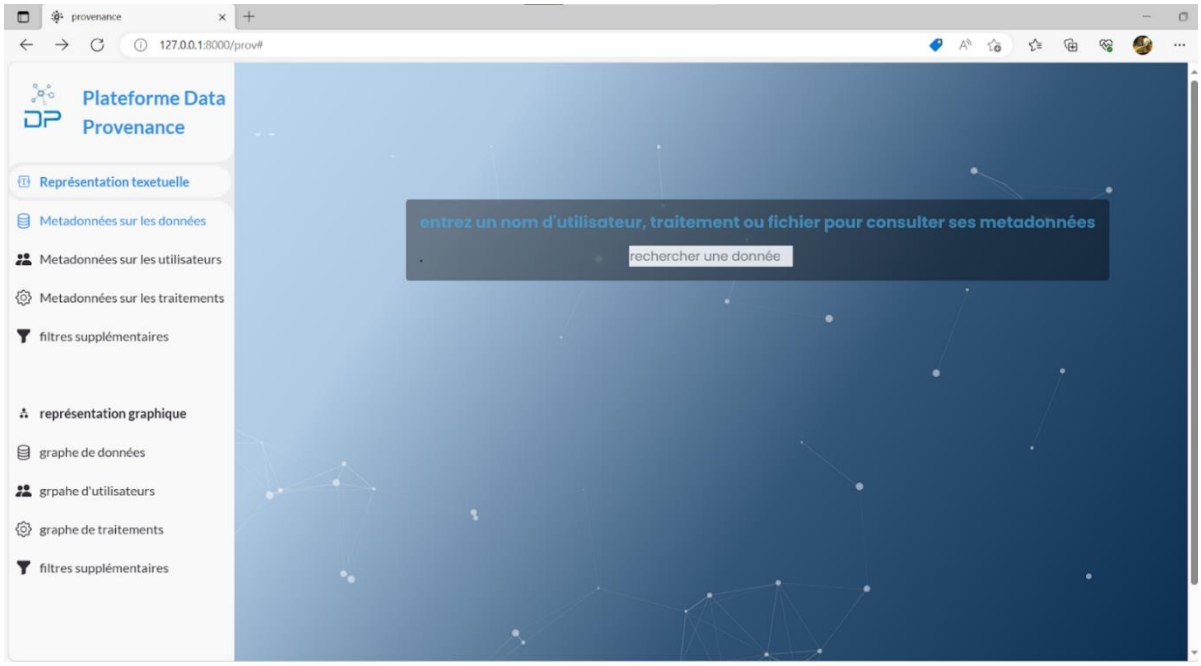


Figure 7. 6: Page d'accueil de l'administrateur.

Pages de visualisation des métadonnées

L'administrateur dispose de deux modes de visualisation selon ses besoins, à savoir :

- La visualisation textuelle : permet d'afficher les métadonnées dans un format textuel.
- La visualisation graphique : permet de restituer les métadonnées dans un format graphique.

En cliquant sur l'une des options de la figure précédente, on obtient l'écran suivant (voir figure 7.7) :

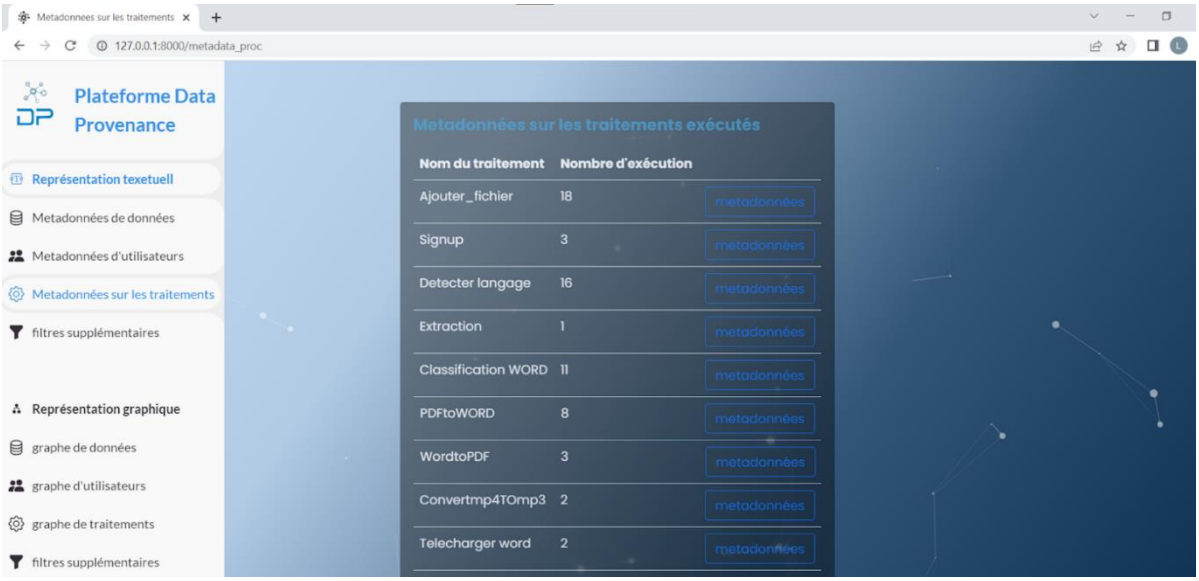


Figure 7.7: Option métadonnées sur les traitements.

Ou cette interface si en clique sur `Métadonnées sur les données`(voir figure 7.8) :

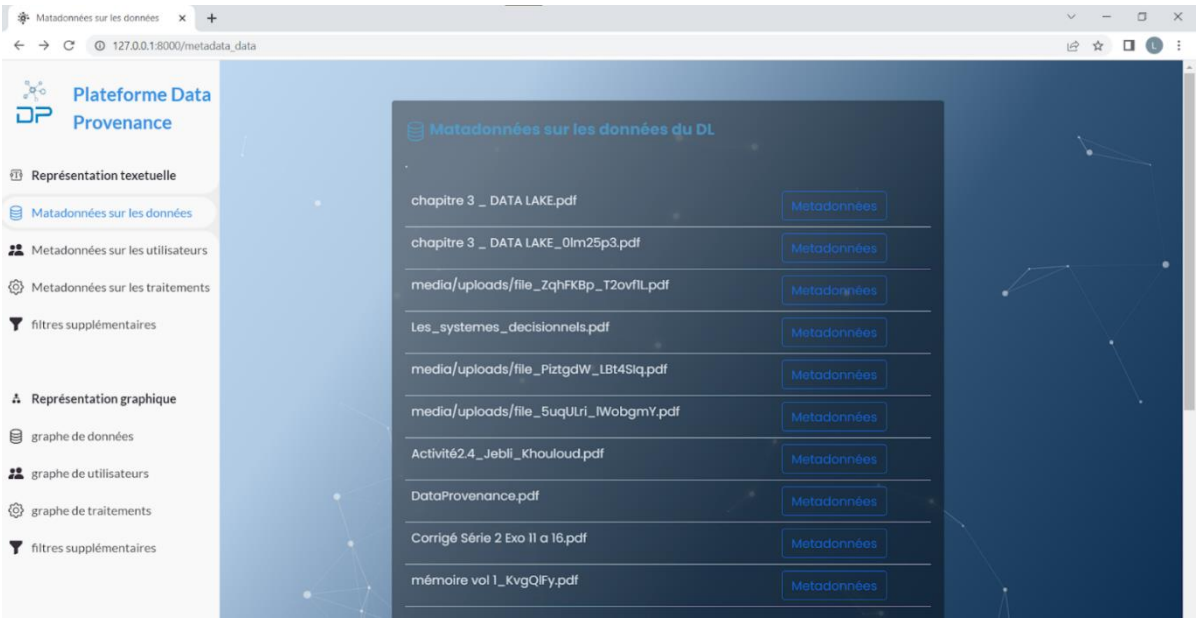


Figure 7.8: Option métadonnées sur les données.

En choisissant ensuite un fichier, ses métadonnées sont affichées comme suit (voir figure 7.9) :

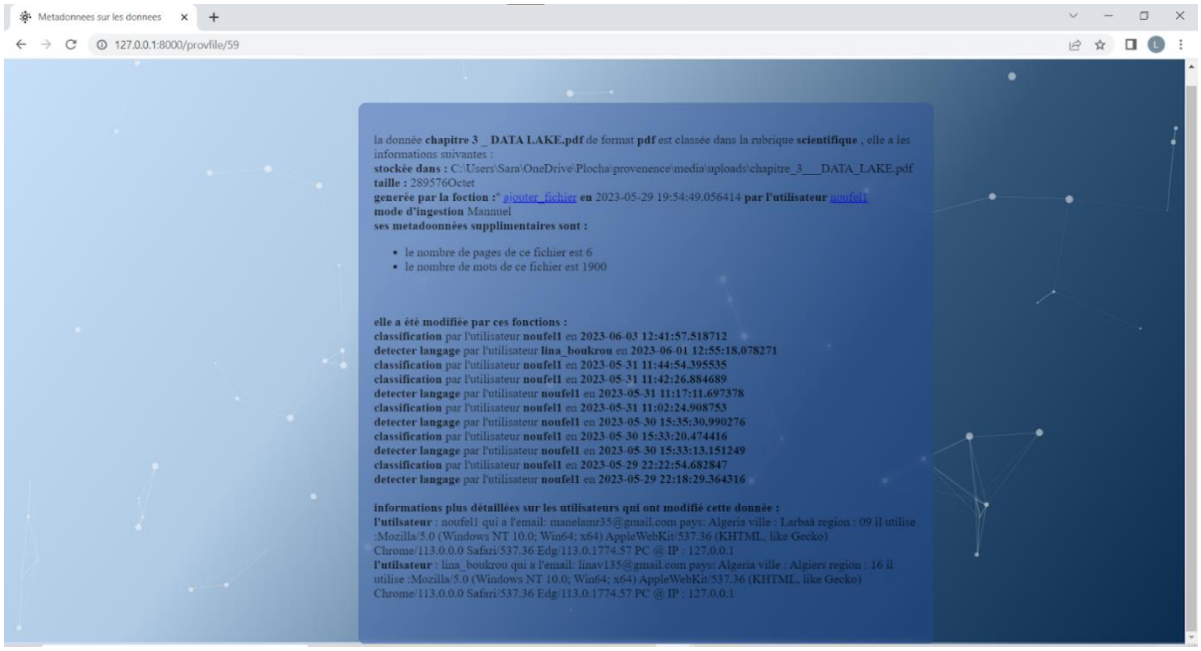


Figure 7.9: Visualisation textuelle de métadonnées d'un fichier.

Ou peut aussi choisir une visualisation graphique comme on peut le voir dans la figure 7.10

Figure 7.10: Graphe de visualisation des données.

Avec ces options, l'ingénieur ou l'administrateur peut effectuer des analyses selon plusieurs dimensions selon ses besoins.

4. Conclusion

Nous avons couvert dans ce chapitre, les technologies utilisées pour le développement de notre plateforme, ses différentes fonctionnalités et les composants dont elle dispose pour la gestion de la data provenance dans un Data Lake.

Conclusion Générale

Notre travail, traite sur le thème de la Data Provenance et la Data Lineage dans un environnement Data Lake. Le big data a induit à l'émergence de bon nombre de problématiques en particulier celles reliées au Lineage de données. Nous avons abordé en premier lieu, les concepts les plus pertinents dans ce domaine à savoir, les big data et leurs caractéristiques, les Data Lake, les métadonnées et leurs rôles ainsi que la data provenance et ses challenges.

Nous avons ensuite fait l'examen de l'ensemble de la littérature relative aux métadonnées, data provenance et Data Lake et avions constitué un état de l'art en étudiant les travaux connexes afin de nous éclairer davantage sur les progrès dans ce domaine. Nous avons identifié à travers cette étape quelques verrous scientifiques ayant motivé notre travail. Bien que les approches courantes aient proposé des systèmes de provenance adaptés aux Big Data, elles n'ont reste pas moins sujettes à quelques limites dont la pauvreté des aspects de provenance abordés. En effet , au niveau de la capture (collecte) , les démarches précédentes se contentent d'informations basiques tels que le nom de la source de données, les noms des personnes (agents) ayant effectué les créations et les modifications des données, aussi , au niveau du stockage, ils ont majoritairement fait usage de bases de données relationnelles telles que MySQL qui a l'inconvénient d'être peu efficace face à une large quantité de données , enfin, l'aspect de la simplicité et de la clarté n'a pas été pris en charge au niveau de la visualisation. Nous avons conclu cette partie par un tableau comparatif et une synthèse.

Concernant notre contribution nous y avons consacré le chapitre 6 dans lequel nous décrivons notre approche en citant les différents composants de notre plateforme ainsi que son mécanisme de capture et de stockage avec des algorithmes. Nous avons également proposé une algèbre de visualisation d'un graphe de provenance.

La troisième partie de notre mémoire est dédiée à la phase d'implémentation de notre application, nous avons introduit les langages de programmation utilisé pour le front end et le back end comme DJANGO, HTML, CSS, JAVASCRIPT, BOOTSTRAP et le SGBD orienté graphe 'Neo4j'. Pour finir, nous avons présenté notre plateforme et les fonctionnalités qu'elle englobe à l'aide de captures d'écran.

En Somme, ce travail n'aura pas été sans contraintes temporelles, techniques et théoriques, ce qui nous a finalement permis d'en apprendre beaucoup sur ce domaine et son importance dans le monde de l'informatique surtout celle des entreprises.

Travaux futurs

Comme tout projet, celui-ci pourra certainement faire l'objet de plusieurs améliorations et perspectives pouvant être résumées comme suit :

- Ajouter plus de formats de données.
- Étendre les traitements vers des traitements ayant une plus grande complexité.
- Intégrer d'autres types de visualisation (tabulaires, par exemple) et ajouter plus de paramètres et de dimensions de visualisation.
- Étendre notre projet en ajoutant d'autres types de métadonnées.

Bibliographie

[1] Gantz, J., & Reinsel, D. (2011). Extracting value from chaos. IDC iview, 1142(2011), 1-12.
[2] Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Hung Byers, A. (2011). Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute.
[3]. Dictionnaire anglais d'oxford : www.oed.com (consulté 26-01-2021)
[4] Sarrakh, R., Suresh, R., Suresh, S., & Al Nabt, S. (2019). Smart solutions in the oil and gas industry: A review.
[5] Doctoral Symposium Paper, A Semantics-enabled approach for Data Lake Exploration Services, Massimiliano Garda, 2019 IEEE World Congress on Services (SERVICES).
[6]Diamantini, C., Giudice, P. L., Musarella, L., Potena, D., Storti, E., & Ursino, D. (2018). A new metadata model to uniformly handle heterogeneous data lake sources. In New Trends in Databases and Information Systems: ADBIS 2018 Short Papers and Workshops, AI* QA, BIGPMED, CSACDB, M2U, BigDataMAPS, ISTREND, DC, Budapest, Hungary, September, 2-5, 2018, Proceedings 22 (pp. 165-177). Springer International Publishing.
[7] F. Castanedo and S. Gidley, 2017, “Understanding Metadata: Create the foundation for a Scalable Data Architecture,” O’reilly.
[8]Sawadogo, P., & Darmont, J. (2021). On data lake architectures and metadata management. Journal of Intelligent Information Systems, 56, 97-120.
[9]Ouahab, R., & Boukraa, D. E. (2021). Développement d'un système de gestion des méta-données dans les data lakes à base de sources NoSQL [Thèse de doctorat, Université de Jijel].
[10]Madsen, M.(2015). How to Build an Enterprise Data Lake: Important Considerations before Jumping In. Third Nature Inc.
[11] Pérez, B., Rubio, J., & Sáenz-Adán, C. (2018). A systematic review of provenance systems. Knowledge and Information Systems, 57, 495-543.
[12] Jagadish, H. V., & Olken, F. (2004). Database management for life sciences research. ACM SIGMOD Record, 33(2), 15-20.
[13] Pinheiro, P., McGuinness, D. L., & McCool, R. (2004). Knowledge provenance infrastructure
[14]Simmhan, Y. L., Plale, B., & Gannon, D. (2005). A survey of data provenance techniques. Computer Science Department, Indiana University, Bloomington IN, 47405, 69.
[15] Miles, S., Groth, P., Branco, M., & Moreau, L. (2005). The requirements of recording and using provenance in eScience experiments (Technical Report). Electronics and Computer Science, University of Southampton.
[16] Galhardas, H., Florescu, D., Shasha, D. E., Simon, E., & Saita, C. A. (2001, June). Improving Data Cleaning Quality Using a Data Lineage Facility. In DMDW (p. 3).
[17]Pérez, B., Rubio, J., & Sáenz-Adán, C. (2018). A systematic review of provenance systems. Knowledge and Information Systems, 57, 495-543.
[18]Aliacar, R., Cabrera, F., Khelifi, M., Mille, A., Palluault, F., Reecht, S., Romain, P., Sibille, C., Stab, G., Teixeira, A., & Vasseur, É. (2015). Traçabilité des données numériques : Quels modèles pour la provenance des données numériques ? État de l'art. Ministère de la Culture et de la Communication.
[19] Dibowski, H., Schmid, S., Svetashova, Y., Henson, C., & Tran, T. (2020). Using Semantic Technologies to Manage a Data Lake: Data Catalog, Provenance and Access Control. In SSWS@ ISWC (pp. 65-80).
[20] Plale, B. (2015). Komadu: A Capture and Visualization System for Scientific Data Provenance.
[21] Suriarachchi, I., & Plale, B. Provenance as Essential Infrastructure for Data Lakes [Preprint, forthcoming in IPAW 2016].
[22] Suriarachchi, I., & Plale, B. (2016, October). Crossing analytics systems: a case for integrated provenance in data lakes. In 2016 IEEE 12th International Conference on e-Science (e-Science) (pp. 349-354). IEEE..

[23] Crawl, D., Wang, J., & Altintas, I. (2011, November). Provenance for mapreduce-based data-intensive workflows. In Proceedings of the 6th Workshop on Workflows in Support of Large-scale Science (pp. 21-30).
[24] Interlandi, M., Wilde, M., Ranganathan, A., Simhadri, A., & Tatbul, N. (2015). Titian: Data provenance support in Spark. Proceedings of the VLDB Endowment International Conference on Very Large Data Bases, 9(3), NIHPublic Access.
[25] Interlandi, M., Ekmekji, A., Shah, K., Gulzar, M. A., Tetali, S. D., Kim, M., ... & Condie, T. (2018). Adding data provenance support to Apache Spark. The VLDB Journal, 27, 595-615..
[26]Castanedo, F., & Gidley, S. (2017). Understanding Metadata. O'Reilly Media, Incorporated. HADI FI PLASAT 9 TA3 6.1

Bibliographie