Big Data entre Qualité & Sécurité de données

Mohamed Talha

UNIVERSITE CADI AYYAD
FACULTE DES SCIENCES ET TECHNIQUES- MARRAKECH

N° d'ordre 19/2022

CENTRE D'ETUDES DOCTORALES
« Sciences de l'Ingénieur »

THÈSE

présentée à la Faculté des Sciences et Techniques de Marrakech

pour obtenir le grade de :

Docteur

Génie Informatique et Mathématiques Appliquées aux Sciences de l'Ingénieur

Spécialité : Informatique

BIG DATA ENTRE QUALITE ET SECURITE DE DONNEES

par :

Mohamed TALHA

(Master : Ingénierie des Systèmes d'Information)

soutenue le 29/10/2022 devant la commission d'examen :

Pr. Said RAKRAK	PES	FST Marrakech,	Président
Pr. Zahi JARIR	PES	FSSM Marrakech,	Examinateur
Pr. Mohamed MOSBAH	PES	INP Bordeaux,	Examinateur
Pr. Habiba CHAOUI	PES	ENSA Kénitra,	Examinatrice
Pr. My. Ahmed EL KIRAM	PES	FSSM Marrakech,	Examinateur
Pr. Mohamed HANINI	PES	FST Settat,	Examinateur
Pr. Anas ABOU EL KALAM	PES	ENSA Marrakech,	Examinateur

Je dédie ce mémoire de thèse à mes parents pour leurs bénédictions, à mon épouse et mes beaux-parents pour leurs soutiens, à mes enfants pour leur inculquer le goût du bon travail et à toute ma famille et mes amis.

Remerciements

Tout d'abord, je remercie Dieu tout-puissant de m'avoir donné le courage, la force et la patience pour mener cette thèse à son terme. La réalisation de ce travail n'aurait pas été achevée sans l'appui de plusieurs personnes qui m'ont aidé, soutenu et encouragé pendant toutes ces années.

Toute ma reconnaissance va à l'endroit du Professeur Anas ABOU EL KALAM pour avoir été encadrant et Directeur de cette thèse. Grâce à la confiance qu'il m'a accordée, à ses conseils avisés, à ses critiques constructives et à son encadrement indéfectible, j'ai pu parfaire ce travail.

Ma pensée va à l'égard du Professeur Nabil ELMARZOUQI, que Dieu ait son âme, avec qui j'ai eu de nombreux échanges bénéfiques pendant les deux premières années de cette thèse.

C'est un grand honneur pour moi que le Jury soit présidé par le Professeur Said RAKRAK à qui j'adresse mes vifs remerciements. Mes sincères remerciements vont à l'égard des Professeurs Habiba CHAOUI, Mohamed MOSBAH et Zahi JARIR pour avoir rapporté cette thèse. Je tiens à exprimer toute ma gratitude aux professeurs Mohamed HANINI, Mohamed MOSBAH, Moulay Ahmed EL KIRAM et Zahi JARIR pour avoir examiné ce travail ; je leur en suis très reconnaissant.

Je ne saurais trouver les mots adéquats pour remercier et exprimer mon témoignage à mon père, à ma mère, à mes beaux-parents et à mon épouse pour leur soutien inconditionnel et leurs encouragements pendant toutes ces années de thèse. Je tiens également à adresser mes remerciements chaleureux à mes frères et sœurs, à ma belle-sœur et à toute ma famille. De toute évidence, je ne peux oublier mes enfants, Arwa et Siraj, qui étaient ma plus grande motivation pour réaliser ce travail.

Mes remerciements s'adressent enfin à tous mes amis et à toutes les personnes qui, de près ou de loin, ont su me donner leur soutien.

Publications et Communications

1. Talha, Mohamed & Abou El Kalam, Anas & Elmarzouqi, Nabil. (2019). Big Data: Trade-off between Data Quality and Data Security. The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019). April 29 - May 2, 2019, Leuven, Belgium. Procedia Computer Science. 151. 916-922. DOI: 10.1016/j.procs.2019.04.127.
2. Talha, Mohamed & Elmarzouqi Nabil & Abou El Kalam, Anas. (2020). Quality and Security in Big Data: Challenges as opportunities to build a powerful wrap-up solution. Journal of Ubiquitous Systems and Pervasive Networks. 12. 09-15. DOI: 10.5383/JUSPN.12.01.002.
3. Talha, Mohamed & Elmarzouqi, Nabil & Abou El Kalam, Anas. (2020). Towards a Powerful Solution for Data Accuracy Assessment in the Big Data Context. International Journal of Advanced Computer Science and Applications. 11. DOI: 10.14569/IJACSA.2020.0110254.
4. Talha, Mohamed & El Mokhtari, Jihane & Abou El Kalam, Anas. (2020). PACT: Privacy Aware Contact-Tracing - Big Data Security and Privacy in the COVID-19 Era. Journal of Multidisciplinary Engineering Science and Technology (JMEST). Volume 7. Issue 5. ISSN: 2458-9403.
5. Talha, Mohamed & Abou El Kalam, Anas. (2021). Big Data between Quality and Security: Dynamic Access Control for Collaborative Platforms. JUCS - Journal of Universal Computer Science 27(12): 1300-1324. DOI: 10.3897/jucs.77046.
6. Talha, Mohamed & Abou El Kalam, Anas. (2021). Big Data: towards a collaborative security system at the service of data quality. The 17th International Conference on Information Assurance and Security (IAS 2021). 14-16 December 2021 on the World Wide Web. HIS 2022, LNNS 420, pp. 1–12, 2022. Hybrid Intelligent Systems. Chapter 55. ISBN: 978-3-030-96304-0. DOI: 10.1007/978-3-030-96305-7_55.

Résumé

Big Data fait référence à de nouvelles technologies et modèles adaptés au stockage et au traitement distribués concernant des données massives et hétérogènes. Dans ce contexte, nous nous intéressons à l'étude de la qualité de données, d'une part, et de la sécurité de données, d'autre part. La qualité de données reflète le niveau de fiabilité que l'on peut assigner aux données. Elle peut être quantifiée à travers de nombreuses dimensions mesurables telles que l'exactitude, la complétude, la fraîcheur et la cohérence de données. La mise en place d'un système de gestion de la qualité de données nécessite généralement une première phase d'évaluation, pour identifier les lacunes, suivie d'une deuxième phase d'amélioration pour corriger ces lacunes. Pour ce travail, nous nous focalisons sur l'étude de la dimension "exactitude" qui mesure le degré auquel les données représentent le monde réel. La sécurité de données, en revanche, s'est traditionnellement penchée sur trois propriétés : la confidentialité, l'intégrité et la disponibilité de données. Nous nous intéressons particulièrement à l'étude de la propriété "confidentialité" qui consiste à protéger les données contre la lecture non-autorisée. Pour cela, nous concentrons notre attention sur le "contrôle d'accès", un des mécanismes les plus répandus pour assurer la confidentialité de données. Dans cette thèse, nous mettons en avant le conflit qui peut avoir lieu entre les processus de gestion de la qualité de données et les mécanismes de sécurité de données. Après avoir donné une vue d'ensemble sur la Big Data et ses technologies et exposé un état de l'art sur la qualité et la sécurité de données, nous présentons notre contribution en trois étapes : tout d'abord, nous proposons un cadre d'évaluation de l'exactitude de données, ensuite, nous exposons un cadre de contrôle d'accès dynamique pour les environnements collaboratifs et distribués, et enfin, nous montrons comment il est possible de se servir d'un système de gestion de la sécurité de données pour améliorer la qualité de données en Big Data. Il s'agit ainsi d'une première étape vers un système réconciliant la qualité et la sécurité de données en Big Data.

Mots clés : Big Data, Contrôle d'accès, Environnements Collaboratifs, Environnements Distribués, Exactitude de données, Objets Connectés, Qualité de données, Sécurité de données.

Abstract

Big Data refers to new technologies and models suitable for distributed storage and processing of massive and heterogeneous data. In this context, we are interested in the study of data quality, on the one hand, and data security, on the other hand. Data quality reflects the level of reliability that can be assigned to the data. It can be quantified through many measurable dimensions such as accuracy, completeness, freshness and consistency of data. The implementation of a data quality management system generally requires a first phase of evaluation, to identify the shortcomings, followed by a second phase of improvement to correct these shortcomings. For this work, we focus on the study of the "accuracy" dimension which measures the degree to which the data represents the real world. Data security, on the other hand, has traditionally focused on three properties: confidentiality, integrity and availability of data. We are particularly interested in the study of the "confidentiality" property which consists in protecting data against unauthorized reading. For this, we focus our attention on "access control", one of the most widespread mechanisms to ensure data confidentiality. In this thesis, we highlight the conflict that can occur between data quality management processes and data security mechanisms. After giving an overview of Big Data and its technologies and presenting a state of the art on data quality and security, we present our contribution in three stages: first, we propose a framework for assessment of data accuracy, then, we expose a dynamic access control framework for collaborative and distributed environments, and finally, we show how it is possible to use a security management system to improve data quality in the Big Data context. This is a first step towards a system reconciling Quality and Security in Big Data.

Keywords: Big Data, Access Control, Collaborative Environments, Distributed Environments, Data Accuracy, Connected Objects, Data Quality, Data Security.

ملخص

تشير البيانات الضخمة إلى مجموعة من التقنيات والنماذج الجديدة التي تعمل على تخزين ومعالجة كميات هائلة من المعلومات غير المتجانسة، و ذلك بشكل موزع. في هذا السياق، نحن مهتمون بدراسة جودة البيانات من ناحية، وأمن البيانات من ناحية أخرى. تعكس جودة البيانات مستوى الموثوقية التي يمكن تخصيصها للبيانات. يمكن قياسها من خلال العديد من الأبعاد القابلة للقياس مثل صحة و اكتمال و حداثة و اتساق البيانات. يتطلب تنفيذ نظام تسيير جودة البيانات، بشكل عام، مرحلة أولى من التقييم، لتحديد أوجه القصور، تليها مرحلة ثانية من التحسين بُغية تصحيحها. في هذا العمل، نركز على دراسة بُعد "صحة المعلومات" الذي يقيس مدى تمثيل البيانات للعالم الحقيقي. من ناحية أخرى، يركز أمن البيانات على ثلاث خصائص ألا و هي سرية و سلامة و توافر البيانات. نحن مهتمون بشكل خاص بدراسة خاصية "السرية" التي تتمثل في حماية البيانات من القراءة غير المسموح بها. لهذا، نركز اهتمامنا على آلية "التحكم في الولوج"، وهي إحدى أكثر الآليات انتشارًا لضمان سرية البيانات. في أطروحة الدكتوراه هذه، نسلط الضوء على التضارب الذي يمكن أن يحدث بين عمليات إدارة جودة البيانات وآليات أمن البيانات. بعد إعطاء نظرة عامة عن البيانات الضخمة وتقنياتها وتقديم أحدث ما توصلت إليه التكنولوجيا بشأن جودة البيانات وأمانها، نقدم مساهمتنا على ثلاث مراحل: أولاً، نقترح إطار عمل لتقييم صحة البيانات، ثم نكشف عن إطار عمل ديناميكي للتحكم في الولوج لبيانات البيئات التعاونية والموزعة، وأخيرًا، نوضح كيف يمكن استخدام نظام إدارة الأمان لتحسين جودة البيانات في سياق البيانات الضخمة. يعتبر هذا العمل خطوة أولى نحو نظام يوفق بين جودة وأمان البيانات الضخمة.

كلمات البحث: البيانات الضخمة، التحكم في الولوج، البيئات التعاونية، البيئات الموزعة، صحة البيانات، الأشياء المتصلة، جودة البيانات، أمن البيانات.

Table des matières

Introduction Générale

Le développement rapide des technologies de l'information, des réseaux sociaux, des appareils connectés et de l'Internet des Objets, contribue à la digitalisation de la vie humaine. La Big Data peut être vue comme une conséquence directe de ces évolutions technologiques. La grande quantité de données que l'on produit aujourd'hui et les nouvelles tendances dans l'exploitation de ces données imposent un vaste changement dans toute la discipline de la technologie conventionnelle. Le terme "Big Data" a été utilisé pour la première fois en 2005 par Roger Mougalas[1] d'O'Reilly Media [Krithika, 20], [Sangeetha, 15]. Il représente aujourd'hui un sujet de recherche préoccupant non seulement pour les universités et les entreprises mais aussi pour les gouvernements. La Big Data fait référence à de nouvelles technologies, de nouvelles architectures et de nouveaux modèles pour la collection, le stockage et le traitement de données. Hadoop, les bases de données NoSQL et le Cloud Computing, entre autres, ont largement participé à l'émergence de la Big Data. La plateforme Hadoop, avec son système de stockage distribué HDFS, son modèle de programmation MapReduce, son gestionnaire de ressources YARN et son large écosystème, fournit un socle solide pour la collection, le stockage et le traitement distribués relatifs à de très gros volumes de données hétérogènes. Les bases de données NoSQL, sous leurs divers types, permettent le stockage et l'analyse des données massives qu'elles soient structurées, semi-structurées ou non-structurées. Le Cloud Computing, de sa part, a permis de fournir les infrastructures nécessaires pour le déploiement de ces environnements distribués avec de nombreux avantages tels que la mise à l'échelle et la multi-location.

La Big Data est souvent caractérisée par trois propriétés : **V**olume, **V**ariété et **V**élocité. Le volume fait référence à de grandes masses de données ; la variété correspond à l'hétérogénéité de données ; la vélocité désigne la rapidité de génération, de collection et de traitement de données. A ces trois propriétés, connues par le modèle 3V, s'ajoutent d'autres propriétés telles que la **V**éracité qui correspond à la fiabilité de données et la **V**aleur qui reflète les profits que l'on peut extraire de ces données. Ces caractéristiques déterminent ainsi les enjeux fondamentaux pour tout projet Big Data.

La qualité de données fait référence au degré auquel un ensemble de caractéristiques de données remplit les exigences [ISO-IEC 25012, 06]. Ces caractéristiques, ou dimensions de qualité, sont des propriétés mesurables qui représentent certains aspects sur lesquels se basent les processus d'évaluation et d'amélioration de la qualité de données. Les dimensions de qualité les plus discutées dans la littérature sont l'exactitude, l'exhaustivité, la cohérence, l'actualité et la pertinence de données [Redman, 96], [Fugini, 02]. Dans ce travail, nous nous intéressons particulièrement à l'étude de l'exactitude de données, une des dimensions les plus complexes à gérer. L'exactitude mesure le degré auquel les données sont correctes. La connaissance préliminaire de cette information permet de déterminer le niveau de confiance que l'on peut attribuer aux données lors des processus de prise de décision. Calculer l'exactitude de données consiste en général à les comparer avec des données de référence que l'on considère correctes. La grande difficulté dans ce processus d'évaluation réside dans l'obtention de ces données de référence. Dans [Talha, 20-b], nous proposons un modèle permettant d'identifier des données de référence grâce à des critères d'exactitude tels que la confiance envers leurs fournisseurs et leurs réputations, l'existence des

[1] https://www.oreilly.com/pub/au/2717

mécanismes de gestion de la cohérence et de la fraîcheur de données, etc. Ce modèle a été ensuite intégré dans un processus complet d'évaluation de l'exactitude de données dans le contexte Big Data.

Par ailleurs, concernant la sécurité de données, il s'agit souvent de les protéger contre les lectures et les écritures non-autorisées et d'assurer leur disponibilité aux entités autorisées. La protection des informations personnelles des individus constitue également un enjeu fondamental pour les projets d'aujourd'hui. La sécurité de données se développe généralement sous une politique (ou une stratégie) qui peut être physique, administrative et/ou logique [Abou El Kalam, 03-a]. La politique de sécurité physique précise un ensemble de procédures et de moyens qui protègent les locaux et le matériel informatique contre des attaques ou des risques physiques (braquage, vol, incendie, inondation, etc.). La politique de sécurité administrative, quant à elle, définit un ensemble de procédures qui traitent la sécurité d'un point de vue organisationnel au sein de l'entreprise (organigramme, répartition des tâches, séparation des environnements de développement et de production, etc.). Enfin, la politique de sécurité logique spécifie les règles d'accès des utilisateurs ; elle est implémentée dans un système de contrôle d'accès. C'est ce dernier type de politiques de sécurité qui nous intéresse pour ce travail. Nous cherchons à mettre en place un système de contrôle d'accès pour protéger les données dans les environnements Big Data. Afin d'orienter ce projet, une étude sur les défis de sécurité, d'une part, et des solutions existantes, d'autre part, est nécessaire. Concernant les défis de sécurité, nous pouvons citer le besoin d'architectures distribuées et évolutives, l'externalisation des infrastructures auprès de tiers externes, le partage des ressources avec des systèmes étrangers, l'hétérogénéité des environnements, des données et de leurs sources, l'intégration des politiques de contrôles d'accès, etc. Quant aux solutions existantes, notamment en termes de contrôle d'accès, de nombreuses approches étendant les modèles classiques basés les rôles (RBAC) et les attributs (ABAC) existent. Dans [Talha, 21-a], nous proposons une extension du cadre PolyOrBAC [Abou El Kalam, 07] destiné à contrôler l'accès dans les environnements collaboratifs en utilisant des services web. Les activités collaboratives sont souvent régies par des agréments d'accès négociés au préalable entre les fournisseurs et les consommateurs de données. Dans PolyOrBAC, la négociation de ces agréments est un processus manuel mené en amont par les acteurs concernés. Cela présente une limitation majeure notamment lorsque la collaboration est réalisée par des objets connectés. Pour automatiser ce processus dans PolyOrBAC, nous nous sommes basés sur la spécification WS-Agreement [Andrieux, 04] qui est aujourd'hui la norme en termes des contrats établis entre les fournisseurs et les consommateurs des services web.

Pour ce travail de thèse, nous nous intéressons particulièrement au conflit qui peut exister entre les systèmes de gestion de la qualité de données et les systèmes de sécurité. Comme expliqué dans [Talha, 19], l'évaluation de la qualité de données nécessite des accès en lecture à de nombreux ensembles de données pour réaliser des opérations de comparaison. Etant dans des environnements distribués, ces ensembles de données peuvent être protégés par des mécanismes de sécurité indépendants. L'amélioration de la qualité de données, quant à elle, peut nécessiter des accès en écriture pour réaliser des opérations de nettoyage et de mise à jour. Pour ce besoin, les règles de sécurité peuvent être encore plus strictes. Cela mène à des situations conflictuelles : faut-il admettre certaines tolérances de sécurité pour fluidifier les processus de gestion de la qualité de données ou faut-il rester ferme d'un point de vue sécurité mais au désavantage de la qualité de données ? Dans [Talha, 21-b], nous abordons cette problématique et nous montrons comment nous pouvons nous servir du cadre PolyOrBAC étendu non seulement pour contrôler l'accès mais aussi pour distribuer les données entre les différents acteurs de l'environnement collaboratif.

Le but étant d'améliorer la qualité de données transitant sur la plateforme et de simplifier les accès pour réaliser des activités collaboratives tout en respectant la politique de sécurité de chaque organisation.

Ce mémoire de thèse est organisé en six chapitres. Le premier chapitre présente une vue d'ensemble sur la Big Data. La première partie de ce chapitre donne une définition de la Big Data et discute ses principales caractéristiques, son importance et ses grands défis. La deuxième partie présente une panoplie de technologies Big Data, en particulier l'environnement Hadoop et certains outils de son écosystème ainsi que les bases de données NoSQL. Bien qu'elle soit technique, cette deuxième partie du premier chapitre est nécessaire à la bonne compréhension de la suite de ce mémoire. Nous présentons ensuite notre état de l'art qui est étalé sur deux chapitres : 2 et 3. Le chapitre 2 étudie la qualité de données en Big Data. Après avoir listé les problèmes de qualité que l'on trouve souvent dans les systèmes d'information, nous présentons quelques modèles et dimensions de la qualité de données ainsi que les techniques de mesure de certaines dimensions. Ensuite, nous présentons les grandes étapes de mise en place d'un système de gestion de la qualité de données ainsi que les techniques les plus utilisées pour améliorer la qualité de données. Nous terminons ce deuxième chapitre par une discussion sur les défis imposés par le contexte Big Data pour gérer la qualité de données. Le chapitre 3 étudie la sécurité de données en Big Data. Nous y discutons les défis de sécurité apportés par le contexte Big Data pour les trois objectifs fondamentaux de la sécurité (confidentialité, intégrité et disponibilité) ainsi que pour la protection de la vie privée des individus. Ensuite, nous présentons les risques de sécurité tout au long du cycle de vie Big Data. Nous dédions une bonne partie de ce chapitre à la présentation des notions fondamentales de la cryptographie (fonctions de hachage, chiffrement, déchiffrement et signature numérique) ainsi que deux concepts fondamentaux qui sont le certificat électronique et la blockchain. Avant d'entamer les systèmes de contrôle d'accès, nous nous arrêtons sur la sécurité de données dans les infrastructures Big Data pour discuter les limites et certaines tentatives de solutions proposées dans la littérature. L'étude du contrôle d'accès est couverte par trois parties : l'étude des modèles de contrôle d'accès classiques, l'étude des langages de contrôle d'accès les plus répandus et, enfin, l'étude du contrôle d'accès en Big Data. Cette troisième partie est organisée selon quatre domaines : les systèmes distribués et collaboratifs, les environnements de stockage sur le Cloud, les environnements intelligents et, enfin, les plateformes Big Data. Notre contribution sur ce travail de thèse est répartie sur trois chapitres : 4, 5 et 6. Le chapitre 4 présente notre cadre d'évaluation de l'exactitude de données. Nous y détaillons la notion de l'exactitude et notre approche pour l'évaluer dans le contexte Big Data (critères d'exactitude, données de référence, échantillonnage de données, mesures de similarité, etc.). Une preuve de notre concept a été développée sur des données ouvertes concernant les stations ferroviaires de la région parisienne en France. Ces données sont offertes en ligne par des organismes gouvernementaux et non-gouvernementaux Français. Le chapitre 5 présente notre cadre de contrôle d'accès dynamique pour les environnements collaboratifs. Ce cadre résulte de l'extension et de l'intégration de deux cadres existants : PolyOrBAC et SLA-Framework. Ce dernier est une implémentation de la spécification WS-Agreement. Après avoir présenté en détail toutes ces solutions, nous montrons comment nous les avons étendues et intégrées ensembles. Le dernier chapitre revient sur la problématique de base à savoir la résolution du conflit entre les systèmes de qualité et de sécurité de données. Nous y présentons notre approche qui nous a permis de nous servir du cadre de contrôle d'accès (présenté dans le chapitre 5) pour alimenter le processus d'évaluation de l'exactitude de données (présenté dans le chapitre 4). Une architecture globale permettant d'améliorer la qualité de données grâce au système de sécurité est ensuite présentée et validée sur une étude de cas, sur les gares ferroviaires de la région parisienne, pour démontrer notre apport.

Chapitre 1. Big Data, une vue d'ensemble

1.1. Introduction

Les données se produisent aujourd'hui à un rythme effréné. De nouvelles unités pour estimer la quantité de données font leur apparition tels que le ZettaByte (10^{21} octets), le YottaByte (10^{24} octets), le BrontoByte (10^{27} octets) ou encore le GeopByte (10^{30} octets). L'IDC (International Data Corporation) prévoit qu'en 2025, la quantité des données produites à l'échelle mondiale atteindra 163 ZettaBytes dont plus de 25% seront créées en temps réel par des objets connectés qui en produisent plus de 95% [Reinsel, 17]. La Big Data est une conséquence directe de cette "digitalisation" de la vie humaine. L'Internet des Objets - IoT (de l'anglais Internet of Things), les objets connectés, les réseaux sociaux et le Cloud Computing ont tous contribué à cette évolution technologique. Les connaissances pouvant être extraites de ces énormes volumes de données forcent les organisations d'y investir pour assurer leur pouvoir concurrentiel. Toutefois, dans le contexte Big Data, de nombreux concepts largement adoptés dans les technologies traditionnelles ne sont plus applicables. A tire d'exemple, la normalisation des données relationnelles n'a plus d'importance en Big Data ; des nouvelles bases de données, dites NoSQL, prennent de plus en plus de place dans les marchés des bases de données et ne tiennent pas en considération les formes normales. Les bases de données NoSQL s'intéressent plutôt au stockage de très gros volumes de données, de différentes structures, de sorte que le traitement de ces données soit rapide et efficace.

Avant l'émergence des technologies Big Data, les entreprises distinguaient entre deux types de données : vivantes ou archivées. Les données vivantes ne concernent que celles nécessaires à l'activité quotidienne de l'entreprise, les autres étaient systématiquement archivées. En outre, les archives ne pouvaient être gardées pendant de longues périodes à cause des limites des solutions traditionnelles de stockage et de leurs coûts élevés ainsi que de la complexité des processus de gestion qui manquaient d'évolutivité et de flexibilité. En revanche, les technologies Big Data ont été conçues pour collecter, stocker, analyser et exploiter de gros volumes de données. Cela a donc permis aux entreprises de profiter pleinement de l'ensemble de leurs données sans avoir recours à l'archivage.

Pour toutes ces raisons, de nombreux projets importants ont été lancés dans différents pays. En 2012, l'Administration de Barack Obama a investi plus de 200 millions de dollars sur un projet de Recherche & Développement en Big Data, intitulé « *Big Data Research and Development Initiative* », dans le but d'améliorer leur compétence à extraire des connaissances et des nouvelles idées à partir de larges collections de données numériques. L'administration était convaincue que ce projet leur promettrait de transformer leur capacité à utiliser les technologies Big Data pour la découverte scientifique, la recherche environnementale et biomédicale, l'éducation et la sécurité nationale [The White House, 12]. Les Nations Unies ont également publié un rapport, intitulé « *Big Data for Development: Opportunities and Challenges* » qui vise à souligner les principales préoccupations concernant les défis de la Big Data et à favoriser le dialogue sur la façon dont elle peut servir le développement international [Global Pulse, 12]. À la suite de ces projets, de nombreux modèles et cadres ont été développés et des nouvelles technologies ont été créées dans le but de fournir plus de capacité de stockage, théoriquement pouvant aller jusqu'à l'infini, des traitements parallèles et des analyses en temps réel des données hétérogènes.

Ce chapitre passe en revue les derniers travaux réalisés en Big Data ainsi qu'un aperçu sur les technologies et les plateformes de stockage et de traitement des données massives. Bien qu'aucune définition standard ne soit encore reconnue pour le terme "Big Data", de nombreux travaux de recherche tentent de le définir selon différents points de vue. La section 2 de ce chapitre est réservée à la définition

de la Big Data et à la discussion de différentes propriétés qui la caractérisent. Dans la section 3, pour démontrer l'importance de la Big Data, nous présentons des exemples d'application dans différents domaines économiques et sociaux. La section 4 présente le cycle de vie des projets Big Data ainsi que les grands défis encore ouverts à la recherche. Pour nous rapprocher davantage de ce domaine relativement nouveau, nous consacrons la section 5 à la présentation des principales technologies et plateformes de traitement des données volumineuses. Nous y présentons la plateforme Hadoop avec son système de stockage de données distribué HDFS et ses systèmes de traitement des données MapReduce et YARN ainsi qu'une sélection d'outils les plus utilisés aujourd'hui qui font partie de l'écosystème Hadoop. Nous y passons également en revue la notion des bases de données NoSQL. La section 5, ayant un contenu purement technique, est très utile à la bonne compréhension des autres chapitres de ce mémoire de thèse, notamment lors de l'implémentation de nos modèles de qualité et de sécurité de données. Enfin, nous concluons ce chapitre dans la section 6.

1.2. Définition, Caractéristiques et Classification de la Big Data

Gartner a défini la Big Data comme étant des sources d'information à haut volume, à haute vélocité et/ou à haute diversité qui nécessitent de nouvelles formes de traitement pour permettre l'amélioration de la prise de décision, la découverte d'informations et l'optimisation des processus [Gartner, 21]. Gantz et Reinsel [Gantz, 11] définissent les technologies Big Data comme étant une nouvelle génération de technologies et d'architectures, conçues pour extraire une valeur économique à partir de très gros volumes de données, en permettant la capture, la découverte et/ou l'analyse à grande vitesse. Chen et Zhang [Chen, 14-a] proposent deux faces de la Big Data. D'un côté, la Big Data est extrêmement précieuse pour produire de la productivité dans les entreprises et des percées révolutionnaires dans les disciplines scientifiques, ce qui pourrait nous donner la possibilité de faire de grands progrès dans de nombreux domaines. De l'autre côté, la Big Data est confrontée à de nombreux défis, tels que les difficultés dans la saisie, le stockage, l'analyse et la visualisation de données. Par ailleurs, les données sont collectées en continu auprès de différentes sources (réseaux sociaux, enregistrements de transactions commerciales de banques, comportements d'utilisateurs sur des sites e-commerces, données de santé, articles publiés sur Internet, …). Cette diversification de données et les contraintes de volumétrie et de vitesse de traitement posent de nombreux challenges à la qualité et à la sécurité de données.

Nous pouvons ainsi définir la Big Data comme étant un ensemble de technologies et de moyens (modèles, techniques, cadres de travail, …) qui permettent de collecter, stocker et traiter, à une grande vitesse, de gros volumes de données hétérogènes dans l'objectif d'en tirer du profit. Ces données peuvent être structurées, semi-structurées ou non-structurées. La tendance aujourd'hui concernant la production de données penche vers des données semi-structurées plus que structurées et des données non-structurées plus que semi-structurées. De nombreux travaux de recherche caractérisent la Big Data par le modèle 5V illustré dans la Figure 1.1.

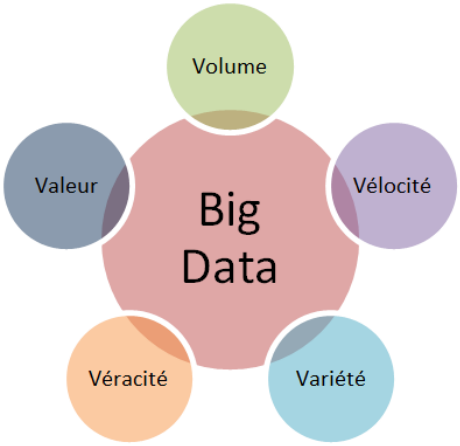

Figure 1.1. Modèle 5V de la Big Data

Le **Volume** fait référence aux énormes quantités de données générées chaque seconde ; le volume de données joue un rôle important dans la mise en œuvre des outils et des modèles de stockage et de traitement de données. La **Vélocité** fait référence à la vitesse à laquelle les données sont produites et traitées. Les données peuvent être importées selon deux modes : les données importées par lots (ou batch), dans lequel des ensembles de données se chargent en une seule fois, et les flux de données (ou streaming), dans lequel les données sont continuellement importées et traitées à la volée. La **Variété** fait référence à l'hétérogénéité de données et de leurs sources. Les données structurées englobent un format de données unifié et peut être facilement géré avec des systèmes de gestion de bases de données relationnelles. En revanche, les données non-structurées et semi-structurées sont plus difficiles à analyser et nécessitent des infrastructures de traitement plus avancées. La **Véracité** fait référence à la cohérence et à la fiabilité des données. Les données peuvent être classées selon le degré de leur qualité (données de bonne ou de mauvaise qualité ou, si on ne peut pas évaluer leur qualité, les classer en données de qualité indéfinie). Enfin, la **Valeur** fait référence à l'utilité des données dans la prise de décision et aux profits que l'on peut tirer de ces gros volumes de données. Il s'agit donc d'une des caractéristiques les plus pertinentes de la Big Data car elle reflète les bénéfices des organisations.

Certains travaux tels que [Katal, 13], [Kaisler, 13], [Gani, 15] et [Khan, 18] considèrent la "**Complexité**" comme étant une sixième caractéristique de la Big Data. La complexité mesure le degré d'interconnexion et d'interdépendance des structures de données de sorte qu'un petit changement, ou une combinaison de petits changements, dans un ou plusieurs éléments, peut produire de très grands changements qui se répercutent dans tout le système et affectent considérablement son comportement. Les auteurs de [Khan, 18] vont plus loin et caractérisent la Big Data par un modèle 10V. Outre les 5V vus jusqu'ici, Khan et al. définissent les caractéristiques suivantes :

- La **Viscosité** reflète la complexité des processus de gestion des données en Big Data.
- La **Variabilité** fait référence à un flux de données incohérent qui peut parfois être affiché.
- La **Volatilité** fait référence à la durée de vie pendant laquelle les données sont valides et pendant combien de temps elles doivent être stockées.
- La **Viabilité** fait référence à la capacité de la Big Data d'être vivante et active ainsi que sa capacité à se développer et à produire plus de données en cas de besoin.
- La **Validité** fait référence à l'exploitation des relations qui peuvent exister entre les données. Un ensemble de données peut ne pas avoir de problèmes de véracité mais peut ne pas être valide s'il

n'est pas correctement compréhensible. De même, un ensemble de données peut être valide pour une application ou une utilisation spécifique et, en même temps, invalide pour une autre application ou utilisation.

Pour mieux comprendre l'ensemble de ces caractéristiques, Hashem et al. [Hashem, 14] classifient la Big Data selon cinq aspects, à savoir les sources de données, le format de données, le stockage de données, la préparation de données et, enfin, le traitement de données.

Concernant le premier aspect, qui est "les sources de données", on trouve des données générées par :

- des médias sociaux qui permettent de partager et d'échanger des informations et des idées sur des communautés et des réseaux virtuels tels que les blogs et les réseaux sociaux.
- des machines qui produisent, de façon automatique, des grandes quantités d'informations à partir du matériel et des logiciels comme des calculateurs et des appareils médicaux.
- des objets connectés qui comprennent, entre autres, les Smartphones, les appareils photo numériques, les tablettes, les équipements intelligents (Télévision, Climatiseur, …). Lorsque ces appareils se connectent les uns aux autres via Internet, ils produisent des processus et des services intelligents qui répondent à des besoins économiques, environnementaux, sanitaires, etc.

Le deuxième aspect concerne "le format de données". Les données peuvent être structurées, semi-structurées ou non-structurées. Les données structurées sont souvent gérées en SQL ; elles sont faciles à saisir, interroger, stocker et analyser à condition que leurs volumes ne dépassent pas les capacités des systèmes de gestion des bases de données relationnelles (SGBDRs). Les données semi-structurées, par exemple des fichiers XML ou JSON, peuvent se présenter sous la forme de données structurées qui ne sont pas organisées en modèles de bases de données relationnelles. La capture de données semi-structurées pour l'analyse est différente de la capture d'un format structuré. Par conséquent, la capture de données semi-structurées nécessite l'utilisation de règles complexes qui décident de manière dynamique du processus à appliquer suite à la réception de nouvelles données. Enfin, les données non-structurées, telles qu'un texte libre ou des fichiers multimédia, sont des données qui ne suivent pas un format spécifié. La collecte et l'analyse de ce type de données constitue aujourd'hui un vrai défi pour les projets Big Data.

Le troisième aspect concerne "le stockage de données". Pour couvrir tous les formats de données, de nouveaux types de bases de données ont été développés que l'on désigne généralement par NoSQL. Différentes catégories existent dans ces bases de données : celles qui stockent les données sous formes de paires "clé-valeur", celles qui regroupent les colonnes dans des colonnes "larges" afin d'optimiser le stockage et le traitement, celles qui stockent les données sous forme de documents, généralement JSON, et enfin, celles qui stockent les données sous forme de graphes centrés sur la gestion des relations entre les objets. L'avantage majeur des bases de données NoSQL réside dans l'absence de schémas fixes pour stocker les données ni de jointures entre les tables ce qui permet un stockage et un traitement distribués de données ainsi qu'une mise à l'échelle plus simple.

Le quatrième aspect concerne "la préparation de données". Cet aspect comporte différents processus tels que :

- le "nettoyage" qui consiste à identifier et à nettoyer des données incomplètes, obsolètes ou de mauvaise qualité.
- la "transformation" qui consiste à transformer les données dans une forme adaptée à l'analyse.

- la "normalisation" qui consiste à structurer les données dans des schémas structurés pour simplifier leur exploitation ou encore minimiser la redondance.

Et enfin, le cinquième aspect concerne "le traitement de données" qui peut être soit par lots (traitement en batch) soit en temps réel (ou quasi-temps réel). Les systèmes basés sur MapReduce de la plateforme Hadoop[2] ont été adoptés par de nombreuses organisations au cours des dernières années pour les travaux par lots de longue durée tandis que les systèmes de traitement en temps réel (ou quasi temps réel) permettent la mise à l'échelle des applications sur des clusters de machines comprenant un grand nombre de nœuds de traitement.

1.3. Ubiquité de la Big Data

Les données aujourd'hui sont générées en grandes quantités, changent rapidement et se présentent sous diverses formes difficiles à gérer et à traiter à l'aide des SGBDRs ou d'autres technologies traditionnelles. Les solutions Big Data fournissent des outils, des méthodologies et des technologies, qui n'étaient auparavant pas disponibles. Ces solutions peuvent être utilisées pour capturer, stocker, rechercher et analyser les données rapidement afin d'extraire des relations et d'en déduire des informations qui peuvent être exploitées pour l'innovation et le gain concurrentiel. Les industries recherchent en continuité de nouvelles et meilleures façons de maintenir leur position et d'être préparées pour l'avenir. La Big Data leur fournit un moyen pour capturer des informations et des idées afin de rester en compétition dans la rude concurrence.

Auparavant, la quantité de données générées n'était pas trop élevée et leur archivage avait pour but d'analyser l'historique des informations (pour des besoins d'audit, de contrôle, de détection d'intrusion, de prévision, de classement, etc.). Aujourd'hui, comme le volume de données se mesure en petaoctets, il n'est plus possible de les archiver et de les récupérer à nouveau pour l'analyse car les besoins ont changé ; les scientifiques de données doivent avoir en main toutes les données pour des analyses prédictives ou de classification, souvent en temps réel, ce qui n'était pas possible auparavant. Nous pouvons constater les avantages des techniques de la Big Data à travers une étude de cas présentée par la banque américaine Zions Bancorporation [Dark Reading, 12]. Cette étude a révélé qu'il fallait entre 20 et 60 minutes pour chercher parmi des données cumulées pendant un mois en utilisant les systèmes SIEM (Security Information and Event Management) traditionnels alors qu'en utilisant son nouveau système Hadoop exécutant des requêtes avec MapReduce et Hive, les mêmes résultats ont été obtenus en environ une minute. L'utilisation avantageuse des techniques de la Big Data telles que les capacités élevées de stockage, de traitement et d'analyse rapide des gros volumes de données non-structurées ne peut qu'améliorer la compétitivité des entreprises.

L'IoT représente l'un des principaux domaines en forte liaison avec la Big Data. D'une part, les objets connectés constituent la principale source générant des données et, d'autre part, les plateformes Big Data permettent d'exploiter efficacement ces grandes quantités de données. Cela a permis aux solutions IoT d'évoluer et d'intégrer de nombreux secteurs sociaux et économiques. Voici quelques exemples d'application IoT/Big Data :

[2] http://hadoop.apache.org/

- **Transport** : il est tout-à-fait possible, grâce à des capteurs installés sur des véhicules, de suivre leurs positions géographiques. La collection et l'analyse en temps réel de ces informations apportent de nombreux avantages aux entreprises et aux particuliers. A titre d'exemples, les entreprises de logistique peuvent superviser et gérer les véhicules, optimiser et adapter les plans de livraison, etc. Les taxis et les entreprises de location des voitures de tourisme avec chauffeurs (VTC) en profitent également en mettant en relation les clients et les chauffeurs, aider les chauffeurs à se diriger vers des endroits où il y a plus de possibilité de trouver des clients, etc. Les particuliers peuvent également en profiter grâce à des appareils SoS (Save our Souls) qui peuvent émettre automatiquement des demandes d'assistance immédiate en cas d'accidents.
- **Santé** : les plateformes de santé connectées (e-santé) permettent à de nombreuses sources de collecter et de transmettre des données sur les individus pour fournir des services personnalisés. Les données des laboratoires, les symptômes des patients téléchargés à partir de capteurs distants ou intégrés, les comptes rendus des opérations hospitalières, les données pharmaceutiques, et bien d'autres, constituent de grands ensembles de données médicales. L'analyse de ces données en temps réel permet d'assister les médecins à adapter des traitements ou d'intervenir rapidement en cas de danger. La sauvegarde de ces données auprès des organisations compétentes permet aux médecins d'urgence de mieux diagnostiquer les problèmes de santé des individus et aux chercheurs de mieux réaliser leurs études dans le respect total de la vie privée des individus.
- **Réseaux Sociaux** : les technologies de la Big Data sont très utilisées dans le domaine des réseaux sociaux aussi bien pour les plateformes qui fournissent des services que pour les organisations intéressées par l'analyse des profils et l'exploitation des informations des utilisateurs enregistrés sur ces plateformes. A titre d'exemple, une entreprise, en connaissant les préférences de ses clients potentiels, peut adapter ses produits afin de s'adresser à un plus grand nombre de personnes. En suivant ce que pensent leurs clients de leurs produits, une entreprise peut avoir un retour d'information lui permettant de mieux s'adapter aux besoins du marché. Un autre exemple d'utilisation de la Big Data dans les réseaux sociaux concerne les élections politiques. La Big Data a joué un rôle déterminant dans la campagne de réélection du président américain Barack Obama en 2012 [InfoWorld, 13], dans les campagnes électorales lors du référendum au Royaume-Uni sur le Brexit ainsi que dans la campagne électorale du président américain Donald Trump pour la présidentielle américaine. Pendant toutes ces compagnes, les intéressés se sont appuyés sur les technologies de la Big Data pour mieux adapter leurs discours et s'adresser directement aux électeurs [Libération, 17].
- **Audit** : les entreprises effectuent souvent l'analyse des audits pour renforcer leurs sécurités, prévoir des plans d'actions ou pour des simples opérations de classification ou de prévision. Par exemple, l'examination des journaux (logs) et l'analyse de l'historique des trafics réseaux et des évènements de sécurité permettent de détecter les anomalies et les intrusions ainsi que de prévenir des cyber-attaques. Les techniques de la Big Data aident non seulement à ne plus avoir besoin de supprimer ces journaux et ces historiques de trafics mais aussi à faciliter leur exploitation afin de renforcer la sécurité des organisations.
- **E-commerce** : les grandes entreprises du commerce électronique traitent chaque jour d'énormes quantités de données dans le but de prendre des décisions commerciales (adapter les prix, passer des commandes, faire des offres spéciales, etc.). Par exemple, eBay utilise des entrepôts de données de plusieurs petaoctets pour gérer les commandes et réaliser des opérations d'analyse sur les recommandations des consommateurs et les transactions commerciales entre les utilisateurs

(vendeurs et acheteurs). Un autre exemple est celui d'Amazon qui a su profiter de la Big Data pour devenir la plus grande plateforme du commerce électronique mondiale. Amazon collecte des données individuelles concernant chacun de ses clients lorsqu'ils utilisent leurs sites web. En plus de ce que le client achète, Amazon garde également une trace des articles consultés, de l'adresse de livraison des utilisateurs ainsi que les avis laissés par les acheteurs. Pour améliorer leurs services, des inventaires sont suivis de façon automatique pour s'assurer que les commandes sont exécutées rapidement. Des modules intelligents permettent de choisir l'entrepôt le plus proche de l'utilisateur, réduisant considérablement les délais d'expédition. Amazon exploite ses données via un moteur de recommandation ; chaque fois qu'un utilisateur cherche un produit spécifique, le moteur de recommandation devine ce qui peut l'intéresser pour le convaincre à l'acheter.

1.4. Cycle de vie des projets Big Data et principaux défis

L'importance de la Big Data réside dans la capacité technique de collecter et d'exploiter efficacement un très grand volume de données quels que soient leurs formats. Grâce à ces techniques, les organisations peuvent avoir une meilleure maitrise de leurs données ce qui permet d'améliorer leurs activités à différentes échelles : mieux se protéger, mieux vendre, mieux s'adapter aux besoins du marché, etc. En revanche, les chercheurs et les professionnels sont confrontés à de nombreux défis et problèmes pour améliorer les mécanismes d'exploitation des données. Les caractéristiques de la Big Data rendent les processus de collecte, de stockage, d'analyse et d'extraction des connaissances très complexes. Dans [Talha, 19] et [Talha, 20-a], nous avons discuté de nombreux problèmes liés à la Qualité et à la Sécurité de données en Big Data. Dans cette section, nous exposons certains des défis les plus discutés dans la littérature pour chaque phase du cycle de vie Big Data.

De nombreux travaux identifient quatre phases principales du cycle de vie d'un projet Big Data. Comme illustré dans la Figure 1.2, la première phase consiste à collecter, de différentes sources, de très gros volumes de données hétérogènes ; la deuxième phase consiste à intégrer toutes ces données sur une plateforme distribuée qui est le lac de données (Data Lake en anglais) ; la troisième phase consiste à analyser ces données pour en déduire des connaissances et des nouvelles idées afin de les exploiter dans la quatrième phase et d'en extraire de la valeur.

Figure 1.2. Cycle de vie des projets Big Data (inspirée de [Alladi, 18] et [Hakuta, 14])

Les défis à relever pendant la phase de collection de données consistent généralement à mettre en place des mécanismes efficaces et des techniques qui peuvent supporter de très gros volumes de données qui se génèrent à une très grande vitesse et par différentes sources. La collection doit se produire dès que les données sont générées, dans des environnements sécurisés, tout en s'assurant de leur véracité et de la fiabilité de leurs sources. Selon le type de données, structurées ou non-structurées, il faut mettre en place

des outils adaptés. Selon les fournisseurs de données (services internes à l'organisation, des organismes certifiés, des objets connectés, des réseaux sociaux, etc.), il faudra déployer tous les mécanismes et aspects techniques et matériels pour garantir la qualité et la sécurité de données.

Concernant la phase de stockage de données, les scientifiques sont souvent confrontés à des problématiques de réduction des coûts et d'optimisation des ressources matérielles et logicielles. Les plateformes de stockage doivent être distribuées au sein d'un même site géographique et/ou sur des sites géographiques différents (souvent dans des pays ou des continents différents afin de minimiser les risques des catastrophes naturelles ou de guerres). La distribution de données pose de nombreux problèmes tels que la gestion de l'intégrité, de la disponibilité et de la redondance de données. En outre, les plateformes de stockage doivent être évolutives de sorte que les processus d'augmentation et de réduction des espaces de stockage doivent être automatiques et ne doivent pas impacter la disponibilité de données. Par ailleurs, la qualité et la sécurité de données constituent deux principaux défis à tenir en considération pour les données d'une entreprise. De nombreux processus pour assurer la qualité de données, tels que le nettoyage et l'intégration de données, doivent être mis en place. Nous reviendrons dans le chapitre suivant sur les problèmes de la qualité de données en Big Data. Pour la sécurité des données, il s'agit d'un des problèmes les plus complexes lors de la phase de stockage de données. En effet, de nombreux problèmes, tels que l'intégration des politiques de contrôle d'accès, le contrôle d'accès à granularité fine, la protection de la vie privée des individus, etc., sont encore ouverts à la recherche.

L'exploitation des données commence à la phase d'analyse qui consiste à comprendre les relations entre les données et y appliquer des algorithmes d'exploration de données pour en extraire des connaissances. L'analyse peut être effectuée en temps réel (sur des flux de données) ou sur des données au repos à travers des traitements par lots. De nombreux domaines, comme le transport, la finance, l'e-santé et l'e-commerce nécessitent de traiter les données en temps réel, ou quasi-temps réel, pour pouvoir prendre des décisions rapides. D'autres applications telles que l'analyse de l'historique des transactions ou des comportements des utilisateurs, l'analyse des journaux, etc. permettent aux organisations d'extraire des informations précieuses et de construire des modèles de données qui peuvent affecter positivement ou négativement leurs positions concurrentielles, surveiller leurs flux d'entrée et de sortie, etc. L'analyse de données doit relever de nombreux défis liés principalement à la nature complexe de la Big Data, y compris les 5V, ainsi que les besoins d'évolutivité, de performances et d'adaptation [Wang, 16], [Tsai, 16].

La dernière phase dans le cycle de vie est l'extraction des connaissances et des nouvelles idées afin de les transformer en des valeurs tangibles. Le tout premier défi à ce niveau réside dans la validité des résultats. Selon si les calculs et les requêtes ont été appliqués sur des données correctes, des données incorrectes ou des données déduites, l'efficacité des résultats change. A cela s'ajoute d'autres problèmes tels que la représentation des données de sortie et leur transformation en des formats prêts à l'exploitation. Par ailleurs, la gouvernance de données constitue un autre défi à tenir en considération lors de l'exploitation de données. Il s'agit de contrôler la conformité des données vis-à-vis de la loi en vigueur, de la transparence et des responsabilités des individus et des systèmes d'information. L'identification des relations entre les données peut mener à la déduction d'informations sensibles ou confidentielles pouvant mettre en danger la vie privée des individus. Jensen dans [Jensen, 13] discute de nombreux défis de la protection de la vie privée dans le contexte Big Data tels que la gestion du consentement des individus, l'obligation par la loi de supprimer les données personnelles, les attaques de ré-identification, etc.

1.5. Technologies Big Data

1.5.1. Apache Hadoop et son Ecosystème

Apache Hadoop[3] est un cadre libre écrit en Java pour le stockage et le traitement distribués de grandes volumétries de données sur des clusters d'ordinateurs. Ces clusters sont construits à partir de matériel de base constitué de simples machines physiques ou virtuelles. Tous les modules de Hadoop sont conçus avec l'hypothèse fondamentale que les pannes matérielles sont courantes et doivent être gérées de façon automatique. Hadoop est aujourd'hui l'outil de-facto utilisé pour l'informatique distribuée. Les idées de bases sur lesquelles a été fondé Hadoop ont été tirées du Système de Fichiers Google GFS (Google File System) tel que présenté dans cet article [Ghemawat, 03] et l'article MapReduce [Dean, 04].

L'avantage clé d'Apache Hadoop est sa conception pour l'évolutivité, c'est-à-dire qu'il est facile d'ajouter des nouvelles machines pour mettre à l'échelle un cluster existant en termes de stockage et de puissance de calcul. En outre, Hadoop part du principe que des machines uniques dans le Cluster peuvent tomber en panne et que leur travail doit être effectué par d'autres machines, au sein du cluster, sans aucune intervention humaine. De cette façon, des clusters énormes, fiables et hautement disponibles peuvent être construits sans investir dans du matériel coûteux.

Le projet Apache Hadoop comprend les modules suivants :

- **Hadoop Common** : un ensemble d'utilitaires utilisés par les autres modules.
- **HDFS** : un système de fichiers pour le stockage distribué de données.
- **MapReduce** : un cadre de traitement distribué de données.
- **YARN** : un gestionnaire des ressources Hadoop.

Les modules énumérés ci-dessus forment le noyau d'Apache Hadoop, tandis que l'écosystème Hadoop contient de nombreux projets tels que HBase, Hive, Spark et bien d'autres que nous passerons en revue par la suite.

1.5.1.1. HDFS

HDFS (Hadoop Distributed File System) est, comme son nom l'indique, un système de fichiers distribué. Il permet d'accéder aux fichiers et aux répertoires stockés sur différentes machines du réseau de manière transparente pour l'utilisateur. HDFS, contrairement à d'autres solutions similaires, se distingue par quelques hypothèses :

- La défaillance matérielle est davantage considérée comme une norme plus qu'une exception. Au lieu de s'appuyer sur des systèmes matériels coûteux et tolérants aux pannes, le matériel de base est choisi. Cela réduit non seulement l'investissement pour la configuration de l'ensemble du cluster, mais permet également le remplacement facile du matériel défaillant.
- Comme Hadoop est conçu pour le traitement par lots de grands ensembles de données, les exigences découlant de la norme POSIX[4], plus centrée sur l'utilisateur, sont assouplies. Par

[3] https://hadoop.apache.org/

[4] POSIX (Portable Operating System Interface) est un ensemble de normes d'interfaces spécifiées par la société informatique IEEE et basées sur UNIX. POSIX fournit la définition des interfaces de programmation d'applications,

conséquent, l'accès à faible latence à des parties aléatoires d'un fichier est moins souhaitable que la diffusion de fichiers à haut débit.

- Les applications utilisant Hadoop traitent des ensembles de données volumineux structurés en fichiers volumineux. Par conséquent, Hadoop est optimisé pour gérer des gros fichiers plutôt qu'une grande quantité de petits fichiers.
- La plupart des applications Big Data écrivent les données une fois et les lisent souvent (fichiers journaux, images, etc.). Par conséquent, Hadoop part du principe qu'un fichier est créé une fois et n'est jamais mis à jour. Cela simplifie le modèle de cohérence et permet un débit élevé.

L'architecture HDFS simplifiée est illustrée dans la Figure 1.3.

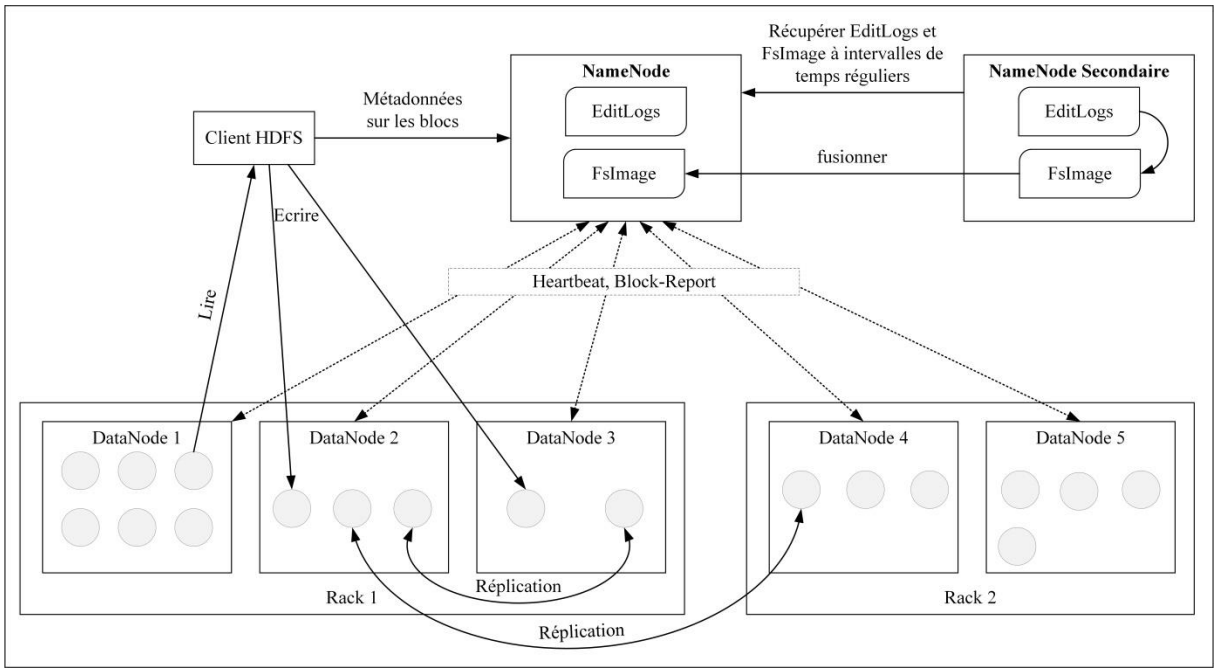

Figure 1.3. Architecture HDFS

Dans HDFS, il existe deux types de serveurs : NameNodes et DataNodes. Le NameNode sert toutes les opérations de métadonnées sur le système de fichiers, telles que la création, l'ouverture, la fermeture ou le renommage de fichiers et de répertoires. Par conséquent, c'est le NameNode qui gère la structure complète du système de fichiers. Concrètement, un fichier est divisé en un ou plusieurs blocs de données qui sont par la suite stockés sur un ou plusieurs DataNodes. Les DataNodes sont rangés dans des racks[5]. La connaissance des blocs de données qui forment un fichier spécifique réside sur le NameNode. Un client HDFS récupère auprès du NameNode la liste des blocs de données constituant un fichier et ensuite contacte les DataNodes pour lire ou écrire des données.

L'ensemble du cluster n'a qu'un seul NameNode ; cela introduit un point de défaillance unique (single point of failure). Deux types de NameNodes existent : le NameNode principal et le NameNode

---

des interfaces utilitaires associées et des Shells de ligne de commande pour la compatibilité des différentes applications et logiciels avec différents systèmes d'exploitation et variantes d'UNIX.

[5] Un rack (ou baie) est une armoire réseau qui regroupe un semble de machines constituant un cluster Hadoop. Ces machines sont réparties sur des racks différents reliés par des commutateurs.

secondaire. La première version de Hadoop ne permettait pas d'exécuter les deux NameNodes en mode failover (basculement automatique). Pendant que le NameNode principal est en panne en raison d'un plantage de la machine ou d'une opération de maintenance, l'ensemble du cluster est hors service jusqu'au basculement manuel vers le NameNode Secondaire ou remise en marche du NameNode principal. A partir de la version 2.0 de Hadoop, il est devenu possible d'exécuter deux instances NameNode dans une configuration active/passive. Comme exigence pour cette configuration, les deux machines exécutant les serveurs NameNodes sont censées avoir un matériel identique et un stockage partagé qui doit être à haute disponibilité.

Toutes les informations sur l'espace de noms HDFS sont stockées sur le NameNode et conservées en mémoire. Les modifications apportées à cette structure de données sont écrites sous forme d'entrées dans un journal de transactions appelé *EditLogs*. Ce dernier est conservé comme un simple fichier dans le système de fichiers du système d'exploitation sur lequel le NameNode s'exécute. L'utilisation d'un journal de transactions pour enregistrer toutes les modifications permet de restaurer ces modifications si le NameNode principal plante. Par conséquent, lorsque le NameNode démarre, il lit le journal de transactions à partir du disque local et applique toutes les modifications qui y sont stockées à la dernière version. Une fois que toutes les modifications ont été appliquées, il stocke la structure de données mise à jour sur le système de fichiers local dans un fichier appelé *FsImage*. Le journal de transactions peut alors être supprimé au fur et à mesure du stockage de ses informations dans le dernier fichier *FsImage*. À partir de là, toutes les modifications ultérieures sont à nouveau stockées dans un nouveau journal de transactions. Le processus d'application du journal des transactions à une ancienne version de *FsImage*, puis de remplacement de sa dernière version est appelé "checkpoint". Actuellement, ces points de contrôle ne sont exécutés que lors du démarrage du NameNode.

La réplication des données est un élément clé de la tolérance aux pannes dans HDFS. Le facteur de réplication d'un fichier détermine le nombre de copies de ce fichier à stocker dans le cluster. La façon dont ces répliques sont réparties sur le cluster est déterminée par la stratégie de réplication. La politique par défaut consiste à stocker une réplique d'un bloc sur le même rack local que l'original et la deuxième réplique sur un autre rack distant. S'il doit y avoir une autre réplique, celle-ci doit être stockée sur le même rack distant que la deuxième réplique. Comme la bande passante du réseau entre les nœuds qui s'exécutent dans le même rack est généralement supérieure à celle entre différents racks (qui doivent passer par des commutateurs), cette politique permet aux applications de lire toutes les répliques du même rack et ainsi de ne pas utiliser les commutateurs réseau qui connectent les racks. L'hypothèse sous-jacente ici est qu'une panne de rack est beaucoup moins probable qu'une panne de nœud.

Les DataNodes stockent les blocs dans leurs systèmes de fichiers locaux. Au démarrage, un DataNode analyse la structure de son système de fichiers local et envoie ensuite une liste de tous les blocs qu'il stocke au NameNode (le Block-Report). Chaque DataNode envoie à intervalles de temps réguliers des signaux de pulsation (heartbeat signal) au NameNode.

HDFS est robuste contre un certain nombre de types de défaillance :

- **Échec du NameNode** : comme le NameNode est un point de défaillance unique, il est essentiel que ses données (EditLog et FsImage) puissent être restaurées. HDFS peut être configuré pour permettre de stocker plus d'une copie des fichiers EditLog et FsImage. Cela peut réduire la vitesse à laquelle le NameNode traite les opérations, mais néanmoins garantit que plusieurs copies des fichiers critiques existent.

- **Échec du DataNode** : si le NameNode ne reçoit aucune pulsation pendant une durée spécifique d'un DataNode, le nœud est considéré comme "mort" et aucune autre opération n'est planifiée pour lui. Un DataNode mort diminue le facteur de réplication des blocs de données qu'il stocke. Pour éviter la perte de données, le NameNode peut démarrer de nouveaux processus de réplication pour augmenter le facteur de réplication pour ces blocs.

1.5.1.2. MapReduce

Alors que HDFS est le système de fichiers distribué pour stocker les données dans Hadoop, MapReduce est le cadre permettant d'analyser ces données. MapReduce est généralement utilisé dans les traitements par lots. Il est hautement évolutif et fiable et est basé sur le principe de « diviser pour régner » qui fournit une tolérance aux pannes et une redondance intégrées. Le principe est de diviser un gros problème en une collection de petits problèmes qui peuvent être résolus rapidement à l'unité. Ce cadre n'exige pas que les données d'entrée soient conformes à un modèle de données particulier ; il peut donc être utilisé pour traiter des ensembles de données non-structurées. Un jeu de données est divisé en plusieurs parties plus petites et des opérations sont effectuées sur chaque partie indépendamment et en parallèle. Les résultats de toutes les opérations sont ensuite résumés pour arriver à la réponse.

MapReduce dispense les développeurs de la gestion des aspects de distribution du travail sur les nœuds du cluster, de récupération des données sur HDFS, de développement spécifique à la couche réseau pour la communication entre les nœuds, etc. Un autre avantage de MapReduce est sa tolérance aux pannes ; si le NameNode remarque qu'un DataNode, à qui il a assigné une tâche, est silencieux pendant un intervalle de temps plus long que prévu, il le considère comme "mort" et assigne la tâche à un autre DataNode. Ceci crée une résilience et facilite le lancement de cette structure logicielle sur des serveurs peu coûteux. Le reste de cette section est basé en grande partie sur le livre "*Big Data Fundamentals - Concepts, Drivers & Techniques*" de Thomas Erl et al. [Erl, 16] qui décrit en détails le fonctionnement interne de MapReduce.

Le moteur de traitement MapReduce fonctionne différemment du paradigme de traitement de données traditionnel qui nécessite le déplacement des données du nœud de stockage vers le nœud de traitement. Cette approche fonctionne bien pour les ensembles de données de petites tailles, mais, avec des ensembles de données volumineux, le déplacement des données peut durer très longtemps. Avec MapReduce, c'est l'algorithme de traitement des données qui est déplacé vers les nœuds qui stockent les données. Cela permet non seulement d'économiser la bande passante du réseau, mais se traduit également par une réduction importante du temps de traitement pour les grands ensembles de données, car le traitement de plus petits morceaux de données en parallèle est beaucoup plus rapide.

Un "Job MapReduce" correspond à l'exécution d'un traitement spécifique dans le moteur de traitement MapReduce. Le Job MapReduce est composée de deux tâches (ou fonctions) : Map et Reduce. Chacune de ces tâches est composée d'un ensemble de sous-tâches comme le montre la Figure 1.4.

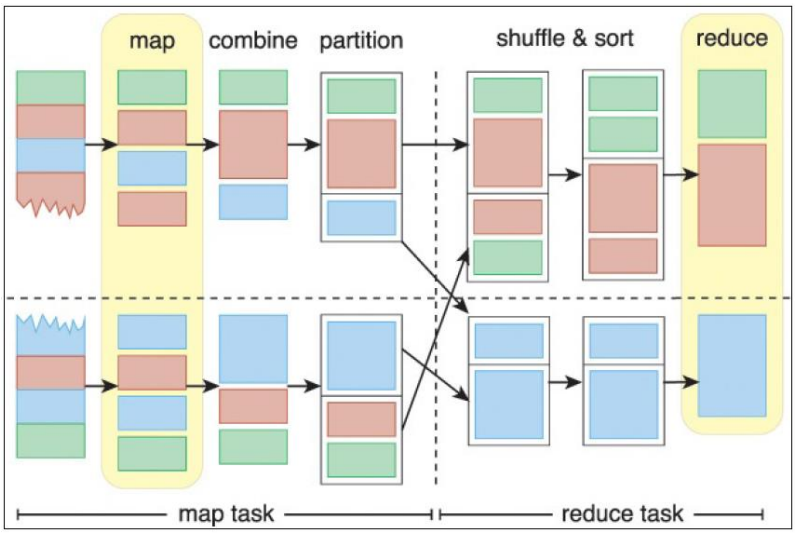

Figure 1.4. Tâches Map et Reduce [Erl, 16]

La première étape de MapReduce est connue sous le nom "Map", au cours de laquelle le fichier de jeu de données est divisé en plusieurs divisions plus petites. Chaque division est composée d'un ensemble d'enregistrements constitutifs sous forme de paires clé-valeur. La clé représente la position ordinale de l'enregistrement, et la valeur contient l'enregistrement lui-même. Les paires clé-valeur analysées pour chaque division sont ensuite envoyées à une fonction de mappage (ou mappeur) qui exécute la logique définie par le développeur (ce qu'il souhaite faire des données). Chaque division contient généralement plusieurs paires clé-valeur, et la fonction de mappage est exécutée une fois pour chaque paire clé-valeur de la division. La fonction de mappage génère en sortie une nouvelle paire clé-valeur pour chaque division. La clé de sortie peut être la même que la clé d'entrée ou une valeur de sous-chaîne de la valeur d'entrée, ou un autre objet sérialisable défini par l'utilisateur. De même, la valeur de sortie peut être la même que la valeur d'entrée ou une valeur de sous-chaîne de la valeur d'entrée, ou un autre objet sérialisable défini par l'utilisateur. Lorsque tous les enregistrements du fractionnement ont été traités, la sortie est une liste de paires clé-valeur où plusieurs paires clé-valeur peuvent exister pour la même clé. Il convient de noter que pour une paire clé-valeur d'entrée, la fonction de mappage peut ne produire aucune paire clé-valeur de sortie comme elle peut en générer plusieurs.

L'étape suivante est la combinaison ou "Combine". Le moteur MapReduce fournit une fonction de combinaison facultative (le combineur) qui résume la sortie de la fonction de mappage (le mappeur) avant qu'elle ne soit traitée par la fonction de réduction (le réducteur). Un combineur est essentiellement une fonction de réduction qui regroupe localement la sortie d'un mappeur sur le même nœud que le mappeur. Le but étant d'éviter de déplacer les données vers le nœud du réducteur. Le combineur peut utiliser la fonction de réduction ou une fonction personnalisée si elle est définie par le développeur. Le moteur MapReduce combine toutes les valeurs d'une clé donnée à partir de la sortie du mappeur, créant plusieurs paires clé-valeur en entrée du combineur où la clé n'est pas répétée et la valeur existe sous forme de liste de toutes les valeurs correspondantes pour cette clé. L'étape de combinaison n'est qu'une étape d'optimisation, et peut donc ne pas être appelée par le moteur MapReduce.

L'étape suivante est le partitionnement ou "Partition". Si plusieurs réducteurs sont impliqués, un "partitionneur" divise la sortie du mappeur ou du combineur (s'il est spécifié et appelé par le moteur MapReduce) en partitions entre les instances de réducteur. Le nombre de partitions sera égal au nombre de

réducteurs. La fonction de partition est la dernière étape de la fonction "Map". Elle renvoie l'index du réducteur auquel une partition particulière doit être envoyée.

Au cours de la première étape de la fonction de réduction, la sortie de tous les partitionneurs est copiée sur le réseau vers les nœuds exécutant la tâche de réduction. Notons que les fonctions de Map et de Reduce s'exécutent forcément sur des nœuds différents ; la copie de la sortie des nœuds de Map vers ceux de Reduce s'appelle "le brassage" (ou Shuffling en anglais). La sortie clé-valeur basée sur une liste de chaque partitionneur peut contenir la même clé plusieurs fois ; le moteur MapReduce regroupe et trie automatiquement les paires clé-valeur en fonction des clés afin que la sortie contienne une liste triée de toutes les clés d'entrée et de leurs valeurs avec les mêmes clés apparaissant ensemble. La manière dont les clés sont regroupées et triées peut être personnalisée par le développeur. Cette fusion crée une seule paire clé-valeur par groupe, où clé représente le groupe et la valeur contient la liste de toutes les valeurs du groupe.

L'étape finale de la tâche de réduction est "Reduce". Selon la logique définie par le développeur, le réducteur résumera davantage son entrée ou émettra la sortie sans apporter de modifications. Dans les deux cas, pour chaque paire clé-valeur qu'un réducteur reçoit, la liste des valeurs stockées dans la partie valeur de la paire est traitée et une autre paire clé-valeur est écrite. La clé de sortie peut être la même que la clé d'entrée ou une valeur de sous-chaîne de la valeur d'entrée, ou un autre objet sérialisable défini par l'utilisateur. La valeur de sortie peut être la même que la valeur d'entrée ou une valeur de sous-chaîne de la valeur d'entrée, ou un autre objet sérialisable défini par l'utilisateur. Tout comme le mappeur, pour la paire clé-valeur d'entrée, un réducteur peut ne pas produire de sortie paire clé-valeur comme il peut en générer plusieurs. La sortie du réducteur, c'est-à-dire les paires clé-valeur, est ensuite écrite dans un fichier séparé (un fichier par réducteur). Le nombre de réducteurs peut être personnalisé.

Le cadre MapReduce, de la version 1.0 de Hadoop, a introduit deux processus essentiels : JobTracker et TaskTracker. Le processus JobTracker s'exécute sur un nœud dédié ou sur le NameNode mais jamais sur un DataNode. Sur chaque DataNode, en général, un processus TaskTracker est lancé. Les fonctions Map et Reduce sont exécutées sur les DataNodes administrés par les TaskTrackers. Chaque DataNode possède un nombre prédéfini de "slots Map" et de "slots Reduce" qui représentent des créneaux d'exécution pour chacune de ces fonctions. Lorsqu'un client émet une demande d'exécution d'un Job MapReduce, c'est le démon du JobTracker qui traite la demande. Il récupère auprès du NameNode l'emplacement des données concernées par le Job et trouve les "meilleurs" nœuds TaskTrackers pour exécuter des tâches en fonction de *(i)* la localité des données, afin de répondre au principe de proximité des données, et *(ii)* des slots disponibles pour exécuter une tâche sur un DataNode donné (l'équilibrage de charge). Le JobTracker surveille les TaskTrackers individuels et renvoie l'état global du Job au client. Chaque TaskTracker se voit attribuer, par le JobTracker, des Job MapReduce à exécuter et reste en communication constante avec le JobTracker pour signaler la progression de la tâche en cours d'exécution. Lorsqu'un TaskTracker ne répond plus, le JobTracker assignera sa tâche à un autre nœud. Lorsque le JobTracker est en panne, HDFS reste fonctionnel mais l'exécution de MapReduce ne pourra pas être démarrée et les tâches MapReduce existantes seront interrompues.

Le fait qu'il n'y a qu'une seule instance de JobTracker dans la version 1.0 de Hadoop a conduit au problème que toute l'exécution de MapReduce peut échouer en cas de panne au niveau du JobTracker (il s'agit d'un point unique de défaillance de MapReduce). En plus, n'avoir qu'une seule instance du JobTracker limite l'évolutivité pour les grands clusters. En outre, le concept des slots prédéfinis pour les

fonctions Map et Reduce a également causé des problèmes de ressources lorsque tous les slots de Map sont utilisés alors que des slots Reduce sont toujours disponibles et vice versa. A cause de tous ces problèmes, il n'était pas possible de réutiliser l'infrastructure MapReduce pour d'autres types de calculs comme les tâches en temps réel. Même si MapReduce est un cadre de traitement par lots, les applications qui traitent des grands ensembles de données, et qui doivent informer immédiatement les utilisateurs des résultats, ne peuvent pas être implémentées avec cette version de Hadoop. Outre le fait que MapReduce 1.0 n'offrait pas de fourniture en temps réel des résultats de calcul, tous les autres types d'applications souhaitant effectuer des calculs sur les données HDFS devaient être implémentés en tant que tâches Map et Reduce, ce qui n'était pas toujours possible.

Pour résoudre ces problèmes, YARN a été introduit dans Hadoop, avec la version 2.0, en tant que gestionnaire de ressources. YARN n'utilise plus de slots pour gérer les ressources. En revanche, les nœuds ont des "ressources" comme la mémoire vive et le nombre de cœurs du processeur qui peuvent être allouées aux applications.

1.5.1.3. YARN

YARN (Yet Another Resource Negotiator) a été introduit avec la version 2.0 de Hadoop en tant que gestionnaire de ressources. La Figure 1.5 illustre l'architecture de YARN.

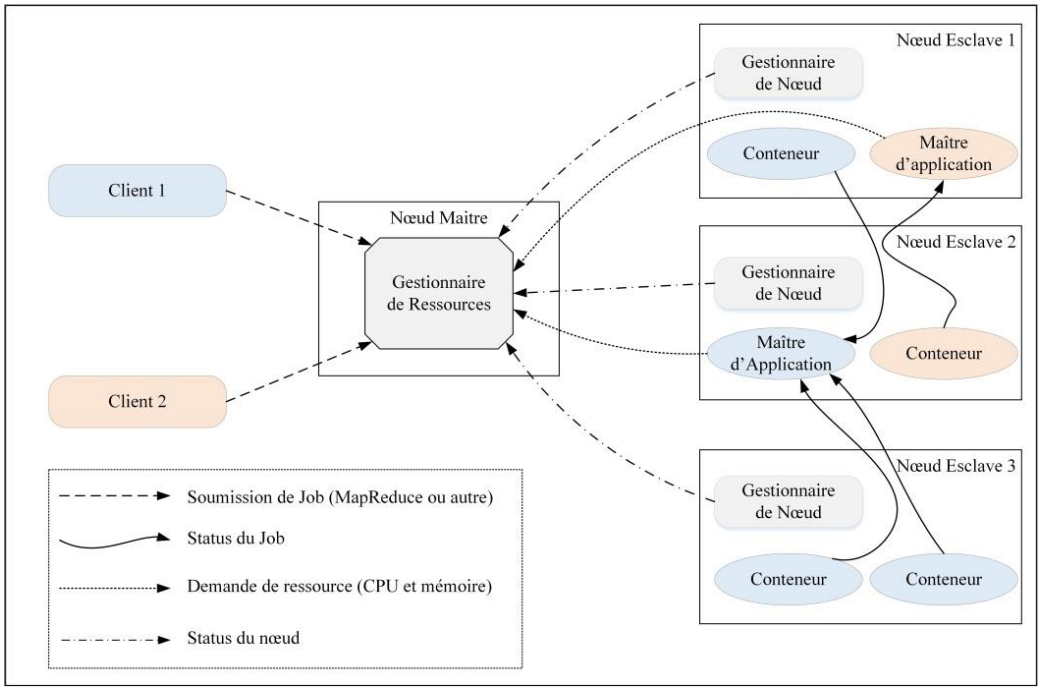

Figure 1.5. Architecture de YARN

Le cœur de YARN est le Gestionnaire de Ressources (Resource Manager) qui s'exécute sur le nœud maître et agit comme un planificateur de ressources global. Il arbitre également les ressources entre les applications concurrentes. Les Gestionnaires de Nœuds (Node Managers) s'exécutent sur des nœuds esclaves et communiquent avec le Gestionnaire de Ressources. Chaque Gestionnaire de Nœud est responsable de la création de conteneurs dans lesquels les applications s'exécutent, surveille leur utilisation du processeur et de la mémoire et les signale au Gestionnaire de Ressources. Chaque

application a son propre Maître d'Application (Application Master) qui s'exécute dans un conteneur et négocie les ressources avec le Gestionnaire de Ressources ; il travaille avec le Gestionnaire de Nœud pour exécuter et surveiller les tâches sur les conteneurs.

L'implémentation MapReduce de Hadoop 2.0 est livrée avec un Maître d'Application, nommée MRAppMaster, qui demande des conteneurs pour l'exécution des tâches Map au Gestionnaire de Ressources. A la réception des identifiants des conteneurs, le MRAppMaster y exécute les tâches Map. A la fin d'exécution de ces tâches, il demande de nouveaux des conteneurs pour l'exécution des tâches Reduce et démarre leur exécution sur les conteneurs fournis.

Si l'exécution d'une tâche échoue, elle est redémarrée par le Maître d'Application. En cas d'échec du Maître d'Application, le Gestionnaire de Ressources tentera de redémarrer l'ensemble de l'application (jusqu'à deux fois par défaut). Par conséquent, le Maître d'Application peut signaler s'il prend en charge la récupération des tâches. Dans ce cas, il reçoit l'état précédent auprès du Gestionnaire de Ressources et ne redémarre que les tâches incomplètes. Si un Gestionnaire de Nœud échoue, c'est-à-dire que le Gestionnaire de Ressources ne reçoit aucune pulsation de sa part, il est supprimé de la liste des nœuds actifs et toutes ses tâches sont traitées comme ayant échoué. Contrairement à la version 1.0 de Hadoop, le Gestionnaire de Ressources peut être configuré pour la haute disponibilité.

1.5.1.4. Ecosystème Hadoop

Hadoop a été développé dans le but de répondre aux besoins fondamentaux de la Big Data, à savoir le stockage distribué, le traitement distribué, le support de tout format de données et la mise à l'échelle. En pratique, cela n'est pas suffisant pour mener des grands projets Big Data. D'autres outils autour de la plateforme Hadoop ont été développés pour répondre à d'autres besoins applicatifs et architecturaux. De nombreux outils de l'écosystème Hadoop bénéficient du stockage distribué (HDFS) et de l'exécution distribuée (MapReduce et YARN) de la plateforme Hadoop.

La liste des outils de l'écosystème Hadoop est longue ; certains ont fait leurs preuves et continuent à évoluer et d'autres n'ont pas pu tenir pour longtemps. Dans cette section nous allons présenter brièvement une sélection des outils les plus utilisés aujourd'hui. Avant cela, nous allons présenter quelques architectures Big Data pour comprendre les relations entre ces outils. Le but étant d'avoir une idée claire sur le fonctionnement de l'écosystème et les interactions entre ses composants plutôt que de présenter un tutoriel complet pour chaque outil. De nombreux travaux dans la littérature, comme [Raj, 18], [Singh, 18], [Oussous, 17] et [Landset, 15], étudient l'écosystème Hadoop et tentent de présenter les avantages et les inconvénients de chaque outil ainsi que les différentes combinaisons possibles.

1.5.1.4.1. Architectures Big Data

Pour mieux expliquer le fonctionnement de l'écosystème Hadoop, nous allons commencer par présenter les architectures les plus appliquées dans les projets Big Data afin de montrer les différentes couches d'une application Big Data ainsi que les tâches dans chaque couche. Cela nous permettra ensuite de mieux comprendre l'utilité de chaque outil et son bon emplacement dans un projet Big Data. Pour cela, nous allons commencer par l'architecture la plus complète qui existe aujourd'hui, à savoir l'architecture "Lambda" qui a été présentée par Nathan Marz et James Warren dans leur livre [Marz, 15]. La Figure 1.6 illustre l'architecture Lambda.

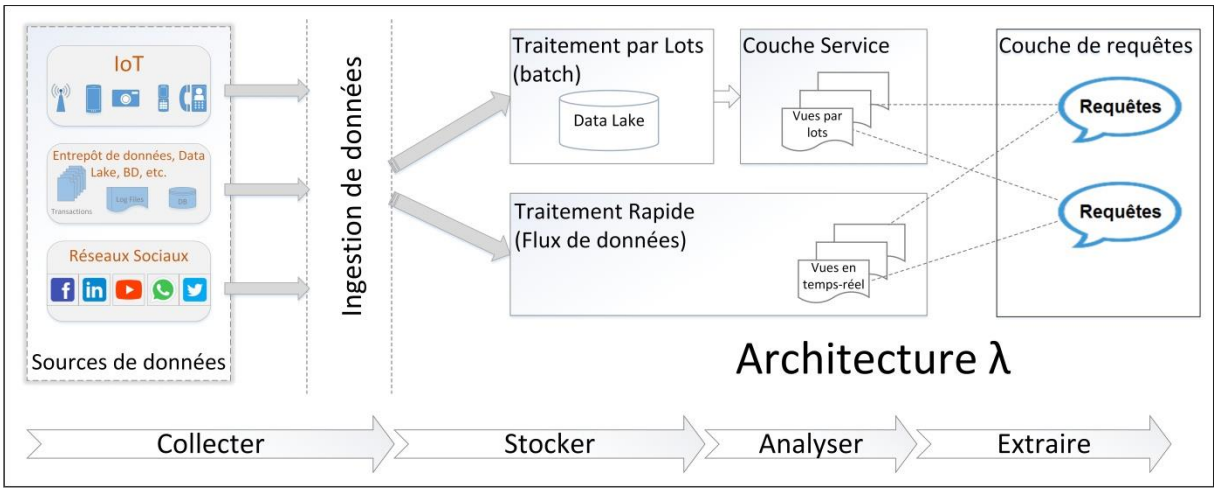

Figure 1.6. Architecture Lambda (inspirée de [Marz, 15])

L'architecture Lambda a pour objectif de répondre aux besoins d'un système distribué robuste et tolérant aux pannes, à la fois contre les défaillances matérielles et les erreurs humaines. Elle est capable de prendre en charge un large éventail de charges de travail et de cas d'utilisation et peut donc être adoptée pour des projets Big Data pour réaliser des traitements de données génériques et évolutifs.

L'architecture Lambda considère différentes sources de données, de tout format, que l'on injecte dans la plateforme Big Data à travers la couche d'ingestion de données. Selon la nature de chaque application, les données seront transmises à la couche de traitement par lots et/ou la couche de traitement rapide. La couche de traitement par lots stocke les données sur le "Data Lake" et leur applique les traitements ensuite ; les résultats seront transmis à la couche de services dans des vues par lots. La couche de service indexe ensuite les vues afin qu'elles puissent être interrogées de manière personnalisée à faible temps de latence. La couche de traitement rapide compense la latence élevée des mises à jour de la couche de service et traite les flux de données en temps réel. Au niveau de la couche de requêtes, il est possible de répondre à toute requête entrante en fusionnant les résultats des vues par lots et des vues générées en temps réel.

Comme il n'est pas toujours indispensable que tous les projets aient une architecture proche de l'architecture Lambda, d'autres alternatives ont fait leurs preuves, en l'occurrence, l'architecture "Kappa". Cette architecture repose sur le principe de fusion de la couche batch (traitement par lots) et la couche temps réel. Cette architecture ne permet donc pas le stockage de manière permanente et est plutôt adaptée pour les projets de traitement des flux de données en temps réel ou quasi-temps réel. L'architecture Kappa présente quatre couches :

- Couche d'ingestion de données.
- Couche de temporisation (buffering) pour la sauvegarde temporaire des données.
- Couche de traitement pour préparer les vues en temps réel.
- Couche d'exploitation.

Concernant les technologies de l'écosystème Hadoop, quelle que soit l'architecture adoptée, pour chaque couche on trouve différents outils qui peuvent être utilisés conjointement ou indépendamment. La Table 1.1 présente une sélection d'outils pour chaque type de tâche.

Table 1.1. Sélection d'outils de l'écosystème Hadoop

Tâche	Exemples d'outils
Ingestion de données	Flume, Sqoop, Kafka
Stockage de données	HBase, Cassandra
Analyse de données	Hive, Pig, Spark, Storm
Exploration de données	Mahout, Drill
Sécurité de données	Ranger, Knox, Sentry
Exploitation et développement	Oozie, Zookeeper

Dans le reste de cette section, nous allons présenter brièvement une dizaine d'outils les plus populaires dans les projets Big Data.

1.5.1.4.2. Principaux outils de l'écosystème Hadoop

1.5.1.4.2.1. Apache Flume, l'ingestion massive des données hétérogènes

Flume[6] est un système distribué conçu pour collecter, agréger et transférer de grandes quantités de données hétérogènes, de diverses sources, vers un magasin de données centralisé tel que HDFS ou HBase. Flume est un outil d'ingestion de données robuste et tolérant aux pannes avec des mécanismes de fiabilité configurables avec de nombreux mécanismes de basculement et de récupération. L'architecture de Flume, illustrée dans la Figure 1.7, est basée sur des flux de données et permet donc une application analytique de données.

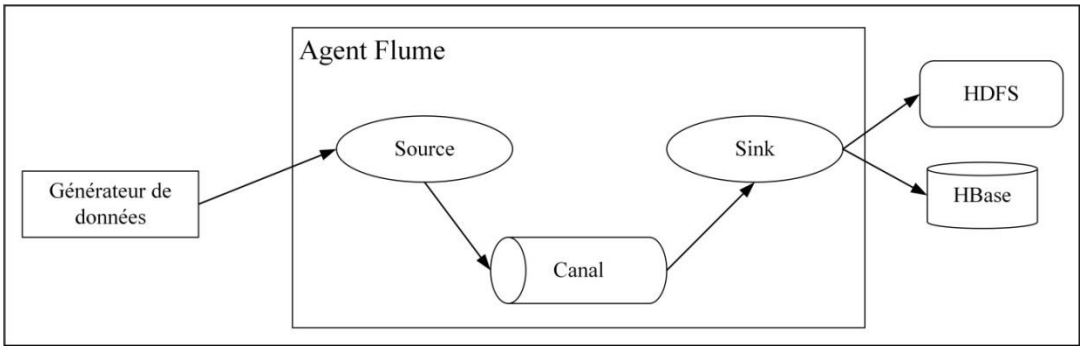

Figure 1.7. Architecture de Flume

L'architecture d'Apache Flume est très simple et flexible. Les générateurs de données, via leurs serveurs web, génèrent des quantités massives de données qui sont collectées, sous forme d'évènements, par des agents individuels, appelés "Agents Flume". Un événement Flume est l'unité de données qui doit être transférée de la source à la destination (un message, un clic sur un site web, une connexion, un nouveau fichier, un paiement, etc.). Un agent Flume est un processus qui tourne sur une machine virtuelle Java ; il reçoit les événements des générateurs de données (ou des autres agents Flume) et les stocke dans des magasins centralisés (HDFS ou HBase). Un Agent Flume contient essentiellement trois composants :

- **Source** : c'est le composant qui reçoit les données des générateurs de données. La source transfère les données reçues vers un ou plusieurs canaux sous forme d'événements.
- **Canal** : c'est le composant qui reçoit les événements de la source et les met en mémoire tampon (buffers) jusqu'à ce que les Sinks les consomment.

[6] https://flume.apache.org/

- **Sink** : c'est un composant qui consomme les données du canal et les stocke dans la destination (qui peut être un magasin centralisé ou d'autres agents Flume).

1.5.1.4.2.2. Apache Sqoop, l'ingestion massive des données relationnelles

Sqoop[7], abréviation de SQL-to-Hadoop, est un système d'import et d'export de données à partir et vers les bases de données relationnelles (Oracle, SQL Server, MySQL, etc.). C'est un canal de transfert de données entre Hadoop et les SGBDRs grâce à des configurations simples. Sqoop permet d'importer et d'exporter toute une base de données ou certaines tables individuelles. Le développeur peut même déterminer quelles colonnes ou quelles lignes seront importées ou exportées. Sqoop utilise la connexion JDBC pour connecter Hadoop à des bases de données relationnelles et peut créer directement des tables dans Apache Hive. Il prend également en charge l'importation incrémentielle si des nouveaux enregistrements ont été ajoutés depuis le dernier import.

1.5.1.4.2.3. Apache Kafka, l'ingestion des flux de données

Kafka[8] gère les flux de données générés par des producteurs de données (des sites web, des applications, des capteurs IoT, etc.). Il s'agit d'un système central qui collecte des données volumineuses, sous forme d'évènements, et les rend disponibles en temps réel pour d'autres applications (des consommateurs). La Figure 1.8 illustre l'architecture Kafka.

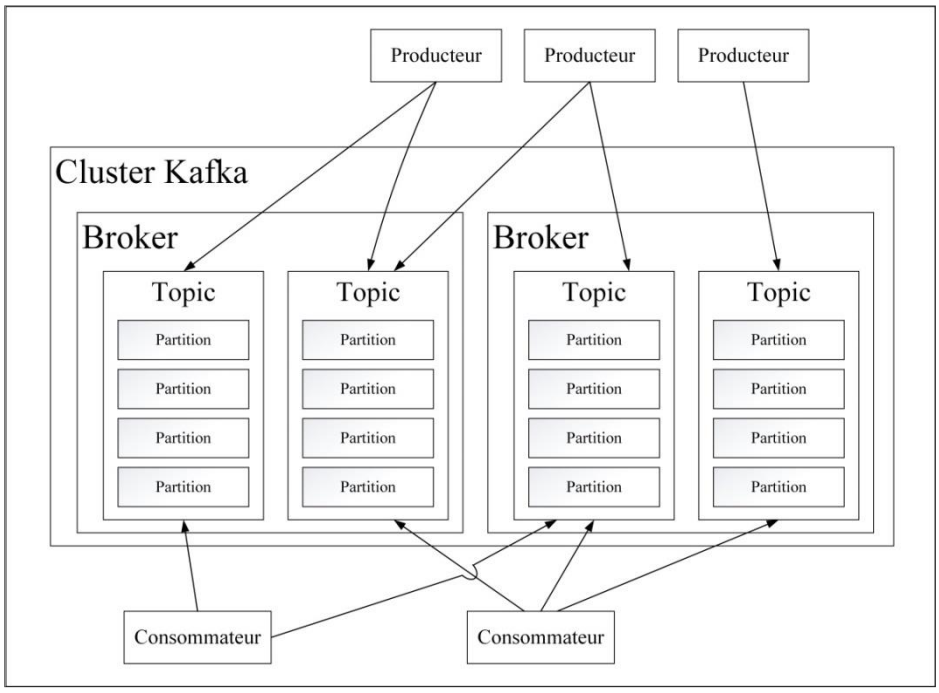

Figure 1.8. Architecture de Kafka

Un Cluster Kafka est composé d'un ensemble de nœuds, appelés des brokers (ou courtiers). Un broker est un composant logiciel composé d'un ou plusieurs topics (ou sujets). Chaque topic est divisé en partitions. La particularité de l'architecture Kafka, c'est qu'une partition peut être placée sur une machine unique ou distribuée sur plusieurs machines pour assurer des écritures ainsi que des lectures en parallèle.

[7] https://sqoop.apache.org/
[8] https://kafka.apache.org/

Le fonctionnement de Kafka consiste à ce que les producteurs publient les événements sur des topics et que les consommateurs récupèrent les évènements sur les topics auxquels ils sont abonnés. Un producteur peut publier sur plusieurs topics et un consommateur peut également s'abonner à plusieurs topics. Les données sont répliquées entre différentes partitions sur différents brokers. Cela assure la fiabilité de Kafka et le rend tolérant aux erreurs ; en cas de problème sur un broker, les informations sont récupérées sur un autre. A la différence des autres outils de messagerie traditionnels tel que Tibco RDV, Tibco EMS ou encore RabbitMQ, Kafka présente la capacité de rétention des messages après leur consommation. La rétention est configurable en fonction de la durée et/ou de la quantité de données que l'on souhaite retenir. Kafka présente également l'avantage d'être horizontalement évolutif ; comme il s'agit d'un Cluster distribué, il suffit d'ajouter des nœuds pour monter en puissance.

1.5.1.4.2.4. Apache HBase, une base de données NoSQL clé-valeur orientée colonnes

HBase[9] est une base de données NoSQL non relationnelle conçue pour fonctionner avec des grands ensembles de données sur HDFS. HBase utilise le modèle clé-valeur ; chaque clé permet d'identifier une valeur unique dans une table très volumineuse sans schéma fixe (les lignes n'ont pas les mêmes attributs). Une table HBase rassemble des données distribuées et persistées sur HDFS sous forme de fichiers spécifiques, appelés HFiles ; HBase bénéficie donc de la tolérance aux pannes fournie par HDFS. La conception d'une table HBase doit tenir en considération cinq éléments :

- **La clé de la ligne (Row-Key)** : chaque ligne d'une table HBase est identifiée de façon unique par une clé, la row-key. La différence entre l'identifiant d'une table relationnelle et la clé d'une table HBase c'est que la row-key est une colonne interne à la structure de la table HBase (comme par exemple les numéros des lignes d'une feuille Excel).
- **La famille de colonnes (Column Family)** : une famille de colonnes représente les valeurs qui doivent être physiquement stockées dans un même ficher HFile. Le concept de "la famille de colonnes" définit la façon dont les données seront persistées et accédées. Lors de la conception de la table HBase, il faut veiller à regrouper dans une même famille, les colonnes qui s'utilisent souvent ensemble (pour éviter d'écrire et de lire dans différents fichiers HFiles). Pour cette raison, HBase est classé parmi les bases de données NoSQL orientées colonnes.
- **Le qualificateur de colonne (Column Qualifier)** : un qualificateur de colonne, ou tout simplement une colonne, est l'équivalent d'une colonne d'une base de données relationnelle. Chaque colonne possède un label qui la qualifie (d'où la nomination qualificateur de colonne). La particularité des colonnes HBase c'est qu'elles sont dynamiques ; certaines valeurs dans une même colonne peuvent être manquantes et c'est cette propriété qui permet à HBase de stocker des données éparses.
- **La cellule (Cell)** : c'est l'intersection entre une row key, une famille de colonnes et une colonne. Une cellule peut contenir des valeurs simples (texte, nombre, date, ...) ou complexes comme des fichiers textes ou multimédia sous forme d'une séquence de bytes (Byte []).
- **L'horodatage (Timestamp)** : tout comme HDFS, les mises à jour consistent à créer des nouvelles versions ; modifier une cellule dans HBase consiste à créer une nouvelle version de la cellule dans laquelle est stockée la nouvelle valeur. La distinction entre les versions se fait grâce à un horodatage assigné à chaque cellule. Par défaut, HBase garde les trois dernières versions de chaque cellule.

[9] https://hbase.apache.org/

L'architecture HBase est composée d'un nœud maître, le HMaster, et d'un ensemble de nœuds esclaves, les RegionsServers. Le HMaster gère les métadonnées des tables HBase et orchestre l'exécution des activités des RegionsServers (pour l'affectation des tâches, l'équilibrage de charge et le traitement des données). Les RegionsServers s'occupent des opérations de lecture/écriture des données dans HDFS (les HFiles sont stockés sur HDFS sous forme de blocs). Pour augmenter la taille du Cluster HBase, il suffit d'ajouter des RegionsServers.

1.5.1.4.2.5. Apache Hive, un entrepôt relationnel pour des données volumineuses

Hive[10] est un entrepôt de données distribué et tolérant aux pannes qui permet des analyses à grande échelle. Il permet aux utilisateurs de lire, écrire et gérer les données stockées sur HDFS avec des simples requêtes SQL grâce au langage HQL. Hive repose sur Hadoop et est donc conçu pour supporter de très gros volumes de données. Il est basé sur un système qui maintient des "métadonnées" qui décrivent les données stockées dans HDFS. Il utilise une base de données relationnelle appelée "metastore" pour assurer la persistance des métadonnées. Une table dans Hive est donc composée de deux éléments : un ensemble de données stockées sur HDFS et leur schéma stocké dans le metastore. Cela permet aux utilisateurs de manipuler les données comme si elles étaient persistées dans des tables relationnelles. Concrètement, Hive convertit les requêtes HQL en jobs MapReduce ou, à partir de la version 0.13 de Hive, jobs Tez qui est un cadre d'exécution sur Hadoop tout comme MapReduce. Cela permet donc aux analystes et développeurs d'exploiter les données HDFS sans se soucier des aspects de programmation des jobs MapReduce (ou Tez). Les interactions entre Hive et Hadoop, comme illustré dans la Figure 1.9, s'effectue en trois étapes :

1. **Requête HQL** : l'utilisateur prépare sa requête HQL sur un client Hive qui peut être un Client Shell (Client Beeline), une application JDBC/ODBC ou une interface web telle que Hue. La requête est ensuite envoyée au serveur Hive et sera traitée par le Service Thrift.
2. **Planification du job** : le pilote Hive s'occupe de la compilation, l'optimisation et la planification du job MapReduce (ou Tez).
3. **Exécution du job** : le job est exécuté sur le cluster Hadoop.

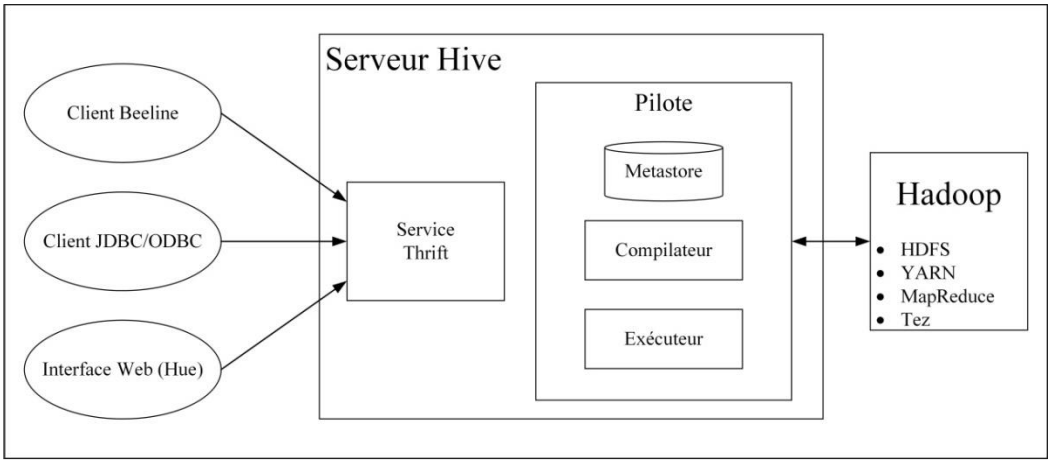

Figure 1.9. Architecture de Hive

[10] https://hive.apache.org/

1.5.1.4.2.6. Apache Pig, l'analyse des données non-structurées

Pig[11] est une plateforme permettant l'écriture des scripts d'analyse de données volumineuses, grâce à son langage Pig Latin, et l'exécution de ces scripts sur un Cluster Hadoop sous forme de jobs MapReduce ou Tez. La différence entre Hive et Pig c'est que le premier permet d'exécuter des requêtes HQL sur des données structurées alors que le deuxième permet d'exécuter des scripts Pig Latin sur des données structurées ou non-structurées. En plus que Pig Latin soit un langage procédural relativement proche à SQL, il est également un langage de flux de données. Cela permet à Pig de disposer d'un ensemble de fonctionnalités pour importer des données sur HDFS ou exporter des données vers des applications tierces. Le cadre Apache Pig englobe différents composants dont les principaux sont illustrés dans la Figure 1.10.

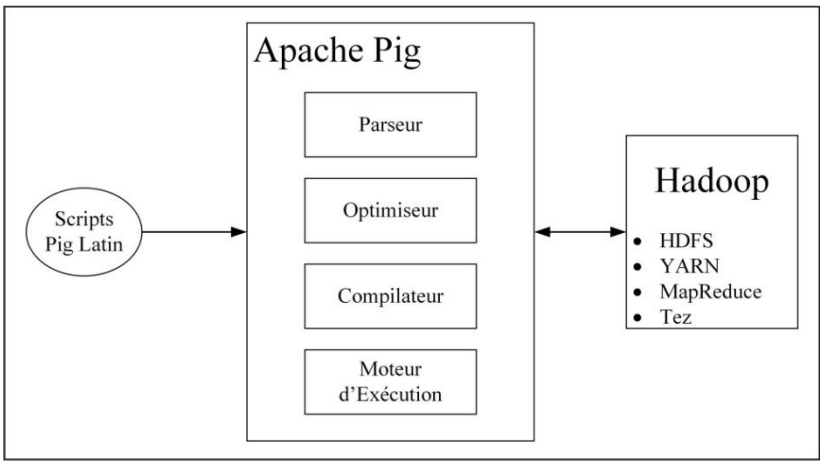

Figure 1.10. Architecture de Pig

Les scripts Pig Latin sont gérés par le Parseur qui est un analyseur syntaxique. Il vérifie, en entrée, la syntaxe de chaque script et, en sortie, il renvoie un graphe acyclique dirigé DAG (Directed Acyclic Graph), qui représente les instructions Pig Latin et les opérateurs logiques entre les instructions. Le DAG est ensuite transmis à l'optimiseur logique qui effectue des optimisations logiques. Le compilateur compile ensuite le DAG optimisé en une série de Jobs MapReduce (ou Tez) qui sont enfin soumis à Hadoop pour exécution.

1.5.1.4.2.7. Apache Spark, le traitement rapide des données massives et des flux de données

Apache Spark[12] est un système de traitement distribué qui peut être utilisé conjointement ou indépendamment de Hadoop. Le point fort de Spark est la mise en cache en mémoire vive et l'exécution optimisée des requêtes sur des données massives. Cela permet à Spark d'être beaucoup plus rapide que MapReduce qui doit absolument passer par des écritures/lectures sur disques (rappelons que les phases Map et Reduce sont exécutées sur des nœuds différents d'où la nécessité de passer par le réseau ou le disque pour transférer les données). Apache Spark peut être utilisé pour réaliser différentes tâches telles que l'exécution du SQL distribué, l'ingestion de données, l'exécution des algorithmes d'apprentissage automatique, le travail avec des graphiques ou des flux de données, etc. Apache Spark regroupe cinq composants principaux :

[11] https://pig.apache.org/
[12] https://spark.apache.org/

1. **Apache Spark Core** : c'est le moteur d'exécution de la plateforme Spark sur lequel reposent tous les autres composants. Il fournit des données de calcul et de référencement en mémoire dans des systèmes de stockage externes.
2. **Spark SQL** : c'est le module qui permet de traiter des données structurées. Les interfaces offertes par Spark SQL fournissent à Apache Spark plus d'informations sur la structure des données et sur le calcul effectué.
3. **Spark Streaming** : ce composant permet à Spark de traiter les flux de données en temps réel. Les données peuvent être chargées de HDFS ou ingérées à travers d'autres sources telles que Kafka ou Flume. Une fois chargées, les données peuvent être traitées à l'aide d'algorithmes complexes et transférées directement vers des systèmes de fichiers ou des bases de données externes.
4. **MLlib** : Apache Spark est équipé d'une bibliothèque riche connue sous le nom de MLlib (pour Machine Learning Library). Cette bibliothèque contient un large éventail d'algorithmes d'apprentissage automatique comme la Classification, la Régression, le Clustering ou encore le Filtrage Collaboratif. Ce module comprend également d'autres outils pour la construction, l'évaluation et le réglage des pipelines de Machine Learning.
5. **GraphX** : Apache Spark est livré avec la bibliothèque GraphX permettant de manipuler et d'effectuer des calculs sur des bases de données graphiques. GraphX unifie le processus ETL (Extract, Transform, Load), l'analyse exploratoire et le calcul itératif des graphes.

Même si le coût de Spark est élevé, car il requiert plusieurs machines pour fonctionner en mémoire, il est considéré comme le moteur de traitement de données le plus efficace, largement utilisé dans les entreprises. De nombreuses caractéristiques le distinguent des autres outils de traitements de données, tels que MapReduce, parmi lesquelles on peut citer :

- **Traitement Rapide** : le point fort d'Apache Spark qui a incité les architectes Big Data à choisir cette technologie est sa vitesse. Les principales unités de données logiques dans Spark sont les RDDs (Resilient Distributed Datasets) qui sont des ensembles distribués d'objets très particuliers. Un RDD peut être divisé en plusieurs partitions logiques qui ont la capacité d'être stockées et traitées sur différentes machines d'un cluster. Le point fort de Spark réside donc dans les RDDs qui permettent de gagner du temps dans les opérations de lecture et d'écriture.
- **Traitement en Temps Réel** : contrairement à MapReduce qui ne traite que les données stockées sur HDFS, Spark est capable de traiter des flux de données en temps réel et est donc capable de produire des résultats instantanés.
- **Bibliothèques Riches** : Apache Spark propose plusieurs bibliothèques pour réaliser des traitements sur des données relationnelles avec SQL, appliquer des algorithmes d'apprentissage automatique, effectuer des analyses complexes sur des graphes, etc.
- **Flexibilité** : même si Spark est lui-même écrit en Scala, les programmes Spark peuvent être écrits avec différents langages : Java, Scala, R ou Python.

1.5.1.4.2.8. Apache Mahout, l'apprentissage automatique

Mahout[13] a été présenté initialement comme une bibliothèque pour l'apprentissage automatique offrant diverses implémentations d'algorithmes de Classification, de Clustering et de Recommandation. Au début, Mahout ne fonctionnait qu'avec Hadoop pour exécuter des algorithmes d'apprentissage automatique sur MapReduce pour des données stockées sur HDFS. Aujourd'hui, il est devenu un cadre

[13] https://mahout.apache.org/

général permettant un mélange de programmation de flux de données et de calculs algébriques linéaires sur des plateformes de traitement rapides comme Apache Spark. Cela a apporté aux utilisateurs la capacité d'exécuter des prétraitements sur les données et la formation des modèles dans un système de flux de données en temps réel.

Apache Mahout présente de nombreuses caractéristiques telles que :

- **Traitement distribué** : les algorithmes de Mahout sont écrits sur Hadoop ce qui leur permet nativement d'être distribués et applicables sur de gros volumes de données.
- **Cadre complet** : Mahout offre au développeur un cadre prêt à l'emploi pour effectuer des tâches d'exploration de données sur de gros volumes de données.
- **Bibliothèque pour l'apprentissage automatique** : Mahout offre une bibliothèque qui comprend plusieurs implémentations de Clustering telles que k-means, fuzzy k-means, Canopy, Dirichlet et Mean-Shift. Mahout prend également en charge les implémentations de classification Naïve Bayes distribuée et complémentaire.

1.5.1.4.2.9. Apache Oozie, la planification des flux de travail

Oozie[14] est un planificateur (Scheduler, en anglais) de flux de travail (Workflow, en anglais) pour Hadoop. Un workflow est une séquence d'actions organisées dans un graphe acyclique dirigé de dépendance de contrôle. Toutes les actions sont en dépendance contrôlée car l'action suivante ne peut s'exécuter que selon la sortie de l'action en cours. Une action de workflow peut être une action MapReduce, Spark, Hive, Pig, Java, Shell, etc. Il peut y avoir des arbres de décision pour décider comment et dans quelles conditions une tâche doit s'exécuter. Il s'agit donc d'un système qui gère le workflow des travaux dépendants, permettant aux utilisateurs de créer des graphiques acycliques dirigés, qui peuvent être exécutés en parallèle ou de manière séquentielle dans Hadoop. Apache Oozie se compose de deux parties :

- Moteur de workflow qui s'occupe du stockage et de l'exécution des tâches des workflows.
- Moteur de coordinateur qui s'occupe de l'exécution des tâches des workflows en fonction des calendriers prédéfinis et de la disponibilité des données.

1.5.1.4.2.10. Apache ZooKeeper, la coordination des services distribués

ZooKeeper[15] est un service distribué largement utilisé dans les plateformes Hadoop pour gérer la coordination d'un grand nombre d'hôtes. Pour sa simplicité de déploiement, ZooKeeper est aujourd'hui le standard pour la gestion et la coordination des grands Clusters. Il est même utilisé dans Apache HBase pour assurer la coordination, la communication et le partage des états entre le HMaster et les RegionsServers. La coordination et la gestion d'un service dans un environnement distribué est un processus complexe. Grâce à son architecture simple et ses APIs, ZooKeeper résout ce problème et permet aux développeurs de se concentrer sur la logique principale des applications.

Concrètement, ZooKeeper peut être déployé dans un cluster pour coordonner les nœuds et maintenir des données partagées avec des techniques de synchronisation robustes. Les services fournis par ZooKeeper sont les suivants :

- **Service de nommage** : ce service assure l'identification des nœuds d'un cluster par leur nom.

[14] https://oozie.apache.org/
[15] https://zookeeper.apache.org/

- **Gestion de la configuration** : ce service transmet à chaque nouveau rejoignant le cluster la configuration nécessaire à jour.
- **Gestion de cluster** : ce service permet de gérer l'ajout ou la suppression des nœuds dans un cluster ainsi que l'état de chaque nœud en temps réel.
- **Élection du chef** : l'élection d'un nœud en tant que "chef" (ou Leader) à des fins de coordination est l'une des principales fonctionnalités sur lesquelles se base ZooKeeper. Il lui permet par exemple d'effectuer une récupération automatique si l'un des nœuds, qu'on appelle les "Followers", tombe en panne.
- **Service de verrouillage et de synchronisation** : ce mécanisme est très utile dans la récupération automatique en cas d'échec d'un nœud.

1.5.2. Bases de données NoSQL

1.5.2.1. Définition

Une base de données NoSQL est un système de gestion de données non relationnel qui, contrairement aux SGBDRs, ne nécessite pas de schéma fixe. Il n'y a donc pas de notion de "jointures" ce qui permet le stockage distribué des données et facilite ainsi la mise à l'échelle. Le terme NoSQL (pour Not only SQL) a été initialement utilisé par Carl Strozzi en 1998. Strozzi suggère que « *comme le mouvement NoSQL actuel s'écarte complètement du modèle relationnel ; il aurait dû être appelé de manière plus appropriée "NoREL"* » [Strozzi, 98].

Un SGBDR traditionnel utilise la syntaxe SQL pour écrire, lire, modifier et supprimer des données. En revanche, un système de base de données NoSQL englobe un large éventail de technologies de base de données qui peuvent stocker des données structurées, semi-structurées, non-structurées et polymorphes. En outre, lorsque le volume de données devient énorme, le temps de réponse d'une base de données relationnelle devient lent, contrairement à une base NoSQL. C'est pour ces deux principales raisons (hétérogénéité et volumétrie) que le concept des bases de données NoSQL est devenu populaire auprès des géants de l'Internet. Une base de données NoSQL répond à quatre caractéristiques :

- **Non-Relationnelle** : il ne doit pas y avoir de jointures entre les tables et il n'y a pas d'obligation que les enregistrements d'une même table aient tous les mêmes attributs.
- **Sans-Schéma** : il ne doit pas y avoir de schémas fixes ou, s'ils existent, ils doivent être assouplis (données semi-structurées).
- **API Simple** : une base de données NoSQL doit offrir des interfaces faciles à utiliser pour le stockage et l'interrogation de données (les plus courants sont les API REST).
- **Distribuée** : le stockage et l'analyse de données dans une base de données NoSQL doivent être distribués sur plusieurs nœuds d'un Cluster. Cela permettra la mise à l'échelle et le basculement automatique (tolérance aux pannes).

1.5.2.2. Types de bases de données NoSQL

Les bases de données NoSQL sont principalement classées en quatre types selon si elles sont basées sur (i) des paires clé-valeur, (*ii*) des colonnes, (*iii*) des documents ou (*iv*) des graphes. Chaque catégorie a ses caractéristiques et ses limites uniques et aucune n'est meilleure pour résoudre tous les problèmes. Le

choix d'une base plutôt qu'une autre est fait en fonction des exigences de chaque projet. Dans la suite de cette section nous présentons brièvement chaque type de bases de données NoSQL ainsi que le théorème CAP, une des principales notions à tenir en considération lors du choix d'une base de données NoSQL.

1.5.2.2.1. Les bases de données NoSQL basées sur une paire "clé-valeur"

Ces bases de données sont conçues pour gérer (lire et écrire) de très gros volumes de données à grande vitesse. Les valeurs sont stockées dans des tables de hachage et sont accessibles via une clé unique. Elles peuvent être de n'importe quel type d'objet binaire (texte, vidéo, document JSON, etc.). Pour assurer l'évolutivité et la disponibilité, les données sont partitionnées et répliquées sur un cluster. Toutefois, ce type de bases de données ne prend souvent pas en charge les transactions. Parmi ces bases de données on trouve Redis[16] et Riak[17].

1.5.2.2.2. Les bases de données NoSQL orientées "colonnes"

Les bases de données orientées colonnes stockent les données dans des tables avec des lignes et des colonnes, tout comme les SGBDRs, mais la différence c'est que les noms et les formats des colonnes peuvent changer d'une ligne à une autre dans une même table. Généralement, ces bases de données regroupent les colonnes associées dans des colonnes "larges" ce qui permet à une requête d'extraire des données associées en une seule opération. Cela permet également d'offrir des performances élevées sur les requêtes d'agrégation, comme COUNT, MIN, MAX, AVG, SUM etc., car les données sont disponibles dans une même large colonne (par exemple, la famille de colonnes présentée précédemment dans HBase). Parmi ces bases de données on trouve HBase et Cassandra[18].

1.5.2.2.3. Les bases de données NoSQL orientées "documents"

Les bases de données orientées documents stockent les données sous forme de documents JSON, BSON (Binary JSON) ou XML. Chaque valeur, qui est un document, est accessible via une clé unique mais peut être également récupéré à travers des requêtes qui cherchent dans le document lui-même. Ainsi, pour permettre une récupération rapide sans connaître la clé, les champs les plus populaires du document doivent être indexés. Les documents peuvent avoir la même structure ou des structures différentes ; au sein d'un même document, les blocs de données peuvent avoir des structures similaires ou différentes. Parmi ces bases de données on trouve MongoDB[19] et CouchDB[20].

1.5.2.2.4. Les bases de données NoSQL orientées "graphes"

Les bases de données orientées graphes utilisent des structures de graphe pour stocker, mapper et interroger des relations entre les données. Les graphes représentent les données sous forme de nœuds, d'arêtes et de propriétés. Les nœuds correspondent aux instances ou aux entités de données qui représentent tout objet à gérer (une personne, un compte, une localisation, un message, etc.). Les arêtes représentent les relations entre les nœuds ; chaque relation a un sens qui peut être unidirectionnel ou bidirectionnel. Chaque nœud et chaque arête ont chacun un identifiant unique. Enfin, les propriétés

[16] https://redis.io/
[17] https://riak.com/
[18] https://cassandra.apache.org/
[19] https://www.mongodb.com/
[20] http://couchdb.apache.org/

représentent des informations descriptives associées aux nœuds et, dans certains cas, les arrêtes. Parmi ces bases de données on trouve Neo4J[21] et OrientDB[22].

1.5.2.3. Théorème CAP

Le théorème CAP, connu également sous le nom du théorème de brasseur, a été introduit par Fox et Brewer [Fox, 99]. Ce théorème indique qu'il est impossible pour un magasin de données distribué d'offrir plus de deux garanties parmi trois : la cohérence C (pour Consistency), la disponibilité A (pour Availability) et la tolérance au partitionnement P (pour Partition Tolerance). La cohérence des données stockées sur un magasin de données distribué exige que tous les utilisateurs voient les mêmes valeurs et en même temps, quel que soit le nœud auquel ils se connectent et quel que soit le nœud ayant réalisé la dernière écriture. La disponibilité exige que tout utilisateur autorisé qui fait une requête obtienne une réponse dans les délais convenus quel soit l'état du système (nœuds non accessibles, maintenance, …). Enfin, la tolérance au partitionnement signifie que le système doit continuer à fonctionner même s'il y a des pannes sur des nœuds de la plateforme distribuée.

Comme illustré dans la Figure 1.11, chaque base de données NoSQL répond à deux propriétés du théorème CAP :

- Cohérence et tolérance au partitionnement : on y trouve HBase, MongoDB et Redis.
- Cohérence et disponibilité : on y trouve OrientDB et Neo4J.
- Disponibilité et tolérance au partitionnement : on y trouve Cassandra, Riak et CouchDB.

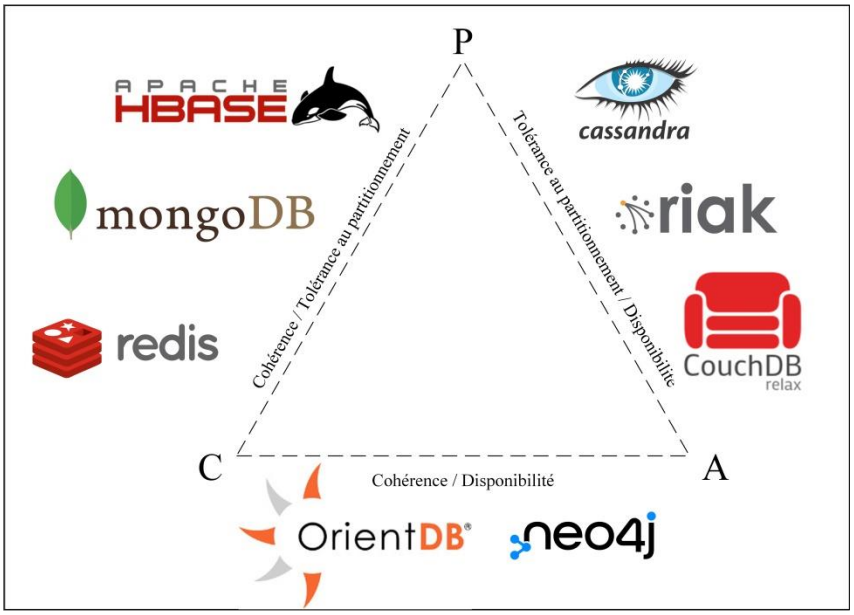

Figure 1.11. Classification des bases de données NoSQL selon le théorème CAP

[21] https://neo4j.com/
[22] https://orientdb.com/

1.6. Conclusion

Les tendances technologiques d'aujourd'hui, telles que l'IoT, les réseaux sociaux et le Cloud Computing, permettent un accès sans précédent à des quantités massives de données hétérogènes. Les architectures et systèmes de gestion des données classiques atteignent leurs limites pour stocker et analyser ces gros volumes de données. De nouvelles approches ont été mises en place pour répondre à de nouveaux besoins tels que le stockage et l'analyse distribués, la mise à l'échelle automatique ainsi que le support de formats de données hétérogènes. Dans ce sens, de nouvelles architectures et technologies, dites Big Data, répondant à ces nouveaux besoins, ont été développées. Tout au long du cycle de vie Big Data, de la phase de collection, au stockage, à l'analyse et jusqu'à la phase d'exploitation de données pour en extraire des connaissances et des idées, on trouve de nouvelles solutions ainsi que des défis résolus et d'autres encore ouverts à la recherche.

Dans ce chapitre, nous avons tenté de donner une définition au terme "Big Data" à travers ses caractéristiques formulés par les modèles 3V, 5V ou encore 10V. Nous avons également présenté les grandes classes de la Big Data à travers cinq aspects : la diversité des sources de données, l'hétérogénéité, le stockage, la préparation et le traitement de données. Nous avons ensuite discuté les principaux défis de la Big Data. Nous reviendrons dans les chapitres sur la Qualité et la Sécurité de données sur plus de détails concernant les défis et les enjeux de la Big Data. Nous avons dédié une grande partie de ce chapitre à la présentation de la plateforme Hadoop et son écosystème ainsi qu'aux bases de données NoSQL afin de fournir un contexte technique sur la façon dont les données sont concrètement ingérées, stockées et analysées aux travers des architectures Big Data. Ce chapitre fournit ainsi une vue d'ensemble sur la Big Data, ses caractéristiques, ses défis et ses technologies, et centralise de nombreuses notions et concepts nécessaires à la bonne compréhension des autres chapitres de ce mémoire de thèse.

Chapitre 2. Qualité de Données en Big Data

2.1. Introduction

La qualité de données représente un sujet essentiel pour les entreprises afin d'obtenir des informations correctes, à jour et précises et d'en prendre en conséquence de bonnes décisions. La qualité de données est un concept multidimensionnel, difficile à caractériser dans des définitions précises même dans le cas de données bien structurées [Firmani, 15]. La Big Data ajoute d'autres dimensions de complexités telles que la volumétrie, l'hétérogénéité et la vitesse de traitement des données. Avec l'utilisation croissante des technologies de l'information et des environnements numériques intelligents, les systèmes d'information interviennent de plus en plus dans des échanges d'informations complexes et opèrent souvent à partir de données obtenues de sources externes, parfois inconnues à priori. Par conséquent, la qualité globale des données qui transitent dans les systèmes d'information peut rapidement se dégrader avec le temps si la qualité des processus et des entrées d'information n'est pas contrôlée [Batini, 09]. La qualité de données doit être définie dans des termes mesurables et clairement définis pour aider une organisation à prendre des mesures basées sur des objectifs tangibles [Suarez, 92]. La qualité a été reconnue par la communauté Big Data comme étant une facette essentielle de sa maturité ; il s'agit d'une pratique cruciale qui doit être mise en œuvre dès les premières étapes de son cycle de vie et progressivement appliquée à travers les autres processus clés [Taleb, 18]. En raison de l'importance des problèmes de qualité, de nombreux travaux de recherche ont été élaborés dans le but de revoir la manière d'évaluer, d'améliorer et de gérer la qualité de données en Big Data [Ge, 18].

Dans un sens général, Crosby définit la "Qualité" comme étant la conformité aux exigences [Crosby, 79]. Suivant ce même principe, le standard ISO/IEC 25012 définit la "Qualité de Données" comme étant le degré auquel un ensemble de caractéristiques de données remplit les exigences [ISO-IEC 25012, 06]. La plupart des processus opérationnels dépendent de la qualité de données ; il s'agit d'un élément clé de l'utilité de l'information tirée des données. Différentes normes s'intéressent à l'étude de la qualité telles que :

- ISO/IEC 25012 [ISO-IEC 25012, 06] qui propose des modèles de qualité englobant un ensemble de caractéristiques que les données de tout système d'information doivent remplir.
- ISO/IEC 25024 [ISO-IEC 25024, 15] qui fournit des mesures générales pour quantifier la qualité de données en se conformant aux caractéristiques de l'ISO/IEC 25012.
- ISO/IEC 25010 [ISO-IEC 25010, 08] qui présente des modèles de qualité pour les logiciels.

Pour pouvoir être traitées et interprétées de manière efficace, les données doivent satisfaire à un ensemble de critères de qualité, communément appelés "Dimensions de Qualité". Les données satisfaisant à ces dimensions peuvent être qualifiées de "bonne qualité". La qualité de données peut être analysée à partir d'une ou plusieurs dimensions selon le contexte de chaque projet. Une dimension est une propriété de qualité de données mesurable qui représente certains aspects (comme l'exactitude, la précision, la consistance, …) qui peuvent être utilisés pour guider les processus de compréhension, d'évaluation et d'amélioration de la qualité.

Les données de mauvaise qualité posent aujourd'hui de nombreux problèmes aux entreprises et aux gouvernements. Selon Saha et al. [Saha, 14], on estime que les données erronées coûtent aux entreprises américaines environ 600 milliards de dollars par an. Les entreprises constatent généralement un taux d'erreur de données d'environ 1% à 5% et, pour certaines entreprises, de plus de 30%. Dans la plupart des projets d'entreposage de données (data warehousing), le nettoyage de données représente 30% à 80% du

temps et du budget de développement pour améliorer la qualité de données. En conséquence, les chercheurs ont soulevé de nombreux défis afin de trouver des modèles et des processus d'évaluation et d'amélioration de la qualité de données en Big Data. Même s'il existe différents modèles pour évaluer la qualité de données dans un contexte classique, aucun de ces modèles n'est adapté aux environnements Big Data [Merino, 15]. En effet, ces modèles sont souvent relationnels, conçus pour manipuler des données dont on maîtrise la structure et ne sont donc pas adaptés à des environnements qui traitent de gros volumes de données, généralement non-structurées et collectées depuis des sources externes. Ainsi, il n'est pas possible d'adopter les approches traditionnelles de la qualité de données dans un contexte Big Data.

Ce chapitre est dédié à une présentation globale de la qualité de données ainsi que les enjeux apportés par le contexte Big Data. Nous y présentons également les principales techniques d'amélioration de la qualité de données et leur application dans des projets Big Data.

2.2. Problèmes de Qualité de données dans les Systèmes d'Information

Des problèmes de qualité de données surviennent lorsque les exigences de qualité ne sont pas satisfaites par les données [Fürber, 11]. Dans le contexte Big Data, la volumétrie et l'hétérogénéité sont les principaux défis empêchant l'application des méthodes et des techniques traditionnelles de gestion de la qualité. Dans les systèmes d'information distribués et coopératifs, les sources de données se caractérisent par divers types d'hétérogénéités qui, selon [Batini, 09], peuvent être classées en 3 catégories :

- **hétérogénéités technologiques** : ce type d'hétérogénéité est dû à l'utilisation de produits de différents fournisseurs et l'emploie de ces produits dans différentes couches d'une infrastructure d'information ou de communication.
- **hétérogénéités de schémas** : ce type d'hétérogénéité est dû principalement à l'utilisation de différents modèles de données (par exemple, une source de données adopte le modèle de données relationnel et une autre source adopte le modèle de données XML) et de différentes représentations pour un même objet (par exemple, une source de données représente un objet « adresse » en tant que table et alors qu'une autre source le représente en tant qu'attribut).
- **hétérogénéités au niveau des instances** : ce type d'hétérogénéité est dû à des valeurs de données différentes et conflictuelles fournies par des sources distinctes pour les mêmes objets. Ce type d'hétérogénéité peut être causé, par exemple, par des processus indépendants et mal coordonnés qui alimentent les différentes sources de données.

L'intégration de données doit faire face à tous les types d'hétérogénéités pour assurer un bon niveau de qualité. Dans cette section, nous nous intéressons à l'étude des différents problèmes impactant la qualité de données selon le modèle d'organisation des données présenté par Oliveira et al. [Oliveira, 05]. Ce modèle organise les données selon quatre niveaux :

- **Sources de données** : à ce niveau, différentes sources de données indépendantes peuvent exister. Il s'agit des générateurs et des fournisseurs de données pour une organisation.
- **Relations multiples** : chaque source de données alimente un ensemble de tables (ou relations) reliées entre elles.
- **Relations simples** : à ce niveau, chaque table (ou relation) est constituée d'un ensemble d'enregistrements (ou tuples).

- **Attributs et tuples** : chaque tuple est composé d'un ensemble d'attributs et chaque attribut possède une valeur dans un tuple.

Dans le reste de cette section, nous allons présenter, sous forme d'exemples, les différents types d'erreurs impactant la qualité des données à chaque niveau d'organisation de données.

2.2.1. Problèmes de qualité de données au niveau des "Sources de Données"

A ce niveau, différents problèmes peuvent impacter la qualité de données, parmi lesquels on cite les exemples suivants:

- Hétérogénéité de syntaxes : par exemple, l'attribut *Date_Insertion* de la relation *Client* de la source de données *DS1* suit le format *jj/mm/aaaa*, alors que l'attribut équivalent de la relation équivalente dans une source de données différente *DS2*, suit le format *mm/jj/aaaa*.
- Hétérogénéité des unités de mesure : par exemple, l'attribut *Prix_Vente* est représenté en euros (€) dans la source de données *DS1*, alors que son équivalent est représenté en dollars ($) dans la source de données *DS2*.
- Hétérogénéité de la représentation : par exemple, l'attribut *Sexe* peut avoir la valeur *F* (pour Féminin) ou *M* (pour Masculin) dans la source de données *DS1*, alors que dans la source de données *DS2* son équivalent peut avoir la valeur *F* (pour Femme) ou *H* (pour Homme).
- Existence de synonymes : par exemple, la relation *Profession* dans *DS1* contient un tuple avec la valeur *Professeur*, alors que l'équivalent de cette relation dans *DS2* contient un tuple avec la valeur *Enseignant*. Les deux valeurs représentent la même profession.
- Existence d'homonymes : par exemple, dans la relation *Produit* de la source de données *DS1*, il existe un produit qui s'appelle *"Souris"* qui est une branche d'une entreprise de vente de matériel informatique, alors que dans la relation *Produit* de la source de données *DS2*, il existe également un produit qui s'appelle *"Souris"* qui est une branche d'une autre entreprise qui vend des animaux domestiques.
- Tuples en double approximatifs : par exemple, le tuple *Client (10, 'John Doe', '118 Avenue de la liberté', 0612345678)* dans la source de données *DS1* est en duplication approximative du tuple *Client (27, 'J. Doe', '118 Avenue de la liberté', 0612345678)* dans la source de données *DS2*. Les deux tuples ont des identifiants différents (10 et 27) et les autres champs sont égaux à part le champ *nom* qui est réduit dans le tuple de la source de données *DS2* (J. au lieu de John).
- Tuples en double incohérents (inconsistants) : par exemple, le tuple *Client (10, 'John Doe', '118 Avenue de la liberté', 0612345678)* dans la source de données *DS1* est en duplication incohérente avec le tuple *Client (27, 'John Doe', '118 Avenue de la liberté', 0687654321)* dans la source de données *DS2* (le numéro de téléphone n'est pas le même).
- Violation de contrainte du domaine métier : par exemple, étant donné que le nombre maximum des groupes de produit est 10. La relation *Groupe_Produit* de la source de données *DS1* contient 7 groupes et la relation *Groupe_Produit* de la source de données *DS2* contient 8 groupes. Si le nombre des groupes de produits distincts résultant de l'intégration (opération d'union des deux ensembles de données) des deux sources de données est supérieur à 10, alors il y aura une violation d'une contrainte métier.
- Utilisation des caractères spéciaux : par exemple, dans la source de données *DS1*, un numéro de téléphone peut avoir la valeur *+212(0)666306108* mais dans la source de données *DS2*, les

parenthèses peuvent être considérées comme des caractères spéciaux empêchant l'intégration des données.

- Différents formats d'encodage : par exemple, dans la source de données *DS1* qui utilise l'encodage **UTF-8**, la valeur *garçon* est correctement renseignée mais si cette valeur est exportée vers une autre source de données qui utilise un encodage non adapté (par exemple **ISO-8859-1**), la valeur sera *gar*  *on* (le caractère *ç* sera remplacé par le caractère).

2.2.2. Problèmes de qualité de données au niveau des "Relations"

A ce niveau, différents problèmes peuvent impacter la qualité des données parmi lesquels on cite les exemples suivants :

- Violation de l'intégrité référentielle : par exemple, l'attribut *Code_Postal_Client* de la relation *Client* contient la valeur *43000*, alors que cette valeur (*43000*) n'existe pas dans la relation *Code_Postal*.
- Référence incorrecte : par exemple, l'attribut *Code_Postal_Client* de la relation *Client* contient la valeur *40000* au lieu de *43000*, alors que ces deux codes postaux (*43000* et *40000*) existent dans la relation *Code_Postal*.
- Hétérogénéité des syntaxes : par exemple, l'attribut *Date_Commande* de la relation *Commande* suit la syntaxe *jj/mm/aaaa*, alors que l'attribut *Date_Facture* de la relation *Facture* suit la syntaxe *mm/jj/aaaa*.
- Circularité entre les tuples dans une même relation : par exemple, un produit peut être un sous-produit d'un autre produit. Le sous-produit est alors enregistré grâce à l'attribut *Code_Sous_Porduit* de la relation *Produit*. Si un produit *X* est enregistré comme étant un sous-produit d'un produit *Y* et que ce même produit *Y* est enregistré comme étant sous-produit du produit *X* alors il y a un cycle entre les tuples dans une même relation.
- Violation de contrainte du domaine métier : par exemple, l'attribut *Total_Facture* de la relation *Facture* contient la valeur *100* alors que le total des valeurs de l'attribut *Valeur_Produit* (pour chacun des produits de la facture) de la relation *Details_Facture* est égal à *90*.

2.2.3. Problèmes de qualité de données au niveau des "Enregistrements" et des "Attributs"

A ce niveau, différents problèmes peuvent impacter la qualité de données parmi lesquels on cite les exemples suivants :

- Valeur manquante : par exemple, l'absence de valeur dans l'attribut obligatoire *Nom* d'un client.
- Violation de syntaxe : par exemple, l'attribut *Date_Commande* contient la valeur *10/11/2012* au lieu de *11/10/2012*.
- Valeur incorrecte : par exemple, l'attribut *Date_Creation* contient la valeur *28/03/2016* au lieu de *29/03/2016*.
- Violation de domaine : par exemple, dans une commande, l'attribut *Quantite_Commandee* contient une valeur négative.

- Violation de contrainte du domaine métier : par exemple, l'attribut *Nom* d'un client doit avoir au moins deux mots (séparés par un espace), cependant, cette contrainte n'est pas respectée dans certains tuples.
- Sous-chaîne invalide : par exemple, l'attribut *Nom_Client* stocke également le diplôme universitaire (par exemple, *Dr. John Taylor*).
- Erreur d'orthographe : par exemple, l'attribut *Adresse* contient la valeur *'Sant Louis'*, au lieu de *'Saint Louis'*.
- Valeur imprécise : par exemple, la valeur *Ant* dans l'attribut *Contact_Client* peut représenter *Anthony*, *Antonio*, *Antonia*, etc.
- Violation de valeur unique : par exemple, deux clients différents ont le même numéro de carte d'identité ou le même numéro de téléphone.
- Existence de synonymes : par exemple, l'attribut *Occupation* contient les valeurs *Professeur* et *Enseignant* dans des tuples différents. Ces deux valeurs représentent la même profession.
- Tuple semi-vide : par exemple, si 60% ou plus des attributs d'un tuple sont vides, alors le tuple est classé comme étant semi-vide.
- Tuple manquant : par exemple, un client qui n'est plus enregistré dans le système suite à une suppression par erreur qui n'a pas été rectifiée.
- Violation de la dépendance fonctionnelle : par exemple, il existe une dépendance fonctionnelle entre deux attributs, *Code_Postal* et *Ville*. Chaque valeur du premier attribut doit être associée à exactement une valeur du second. Par conséquent, les valeurs suivantes de deux tuples violent la dépendance fonctionnelle: (*Code_Postal = 4000* ; *City = "Porto"*) et (*Code_Postal = 4000* ; *City = "Lisboa"*).
- Valeur Ambiguë : le manque de précision peut être considéré comme une ambiguïté (si, par exemple, dans un hôpital, les patients sont identifiés par *Nom* et *Prénom*, deux personnes ayant les mêmes noms et prénoms peuvent créer un problème d'ambiguïté).
- Valeur temporelle périmée : par exemple, une personne est enregistrée comme étant père d'un seul enfant. Si sa situation change (avoir d'autres enfants) et que sa fiche ne se met pas à jour, alors l'information sera périmée (elle était correcte dans le passé mais elle ne l'est plus).
- Valeurs contradictoires : par exemple, si une personne est titulaire d'un permis de conduire et que sa fiche n'a pas été mise à jour suite à la perte de son permis (évolution temporelle). Ce cas représente une situation contradictoire.
- Mauvais type de données : par exemple, la date d'une commande qui est enregistrée sous forme de chaînes de caractères.
- Entrée fictive: par exemple, lors de la phase de test d'une application, des valeurs fictives ont été saisies et que ces mêmes valeurs ont été déployées en Production.

2.2.4. Classification des problèmes de qualité de données

Comme nous avons pu le voir, tous les niveaux confondus, ces problèmes sont étroitement liés et devraient donc être traités de manière uniforme en suivant un processus bien déterminé de nettoyage de données. Une classification des problèmes s'avère nécessaire pour atteindre un tel objectif. Müller et Freytag [Müller, 03] proposent une classification des problèmes de qualité selon trois types d'anomalies : syntaxique, sémantique ou de couverture (manque de valeur) :

- **Les anomalies syntaxiques** : elles comprennent des erreurs lexicales (divergences entre la structure des éléments de données et le format spécifié), des erreurs de format de domaine (erreurs lorsque la valeur assignée à un attribut donné n'est pas conforme au format de domaine prévu) et des irrégularités concernant l'utilisation non uniforme de valeurs (par exemple, l'utilisation de différentes devises).
- **Les anomalies sémantiques** : elles comprennent les violations des contraintes d'intégrité (quand des tuples ou des ensembles de tuples ne satisfont pas à une ou plusieurs des contraintes d'intégrité), les contradictions (des valeurs dans un tuple ou entre des tuples violent une sorte de dépendance entre les valeurs ; par exemple, un écart entre l'âge et la date de naissance), les entrées dupliquées et les tuples invalides.
- **Les anomalies de couverture** : elles concernent les valeurs et les tuples manquants.

En suivant ce raisonnement, nous pouvons classer les différents problèmes listés dans cette section dans la Table 2.1.

Table 2.1. Classification des problèmes impactant la qualité de données

Niveau de Stockage	Types d'anomalie		
	Syntaxique	Sémantique	de Couverture
Sources de données	◊ hétérogénéités de schémas ◊ Hétérogénéité de syntaxes ◊ Hétérogénéité des unités de mesure ◊ Hétérogénéité de la représentation ◊ Utilisation des caractères spéciaux ◊ Différents formats d'encodage	◊ Existence de synonymes ◊ Existence d'homonymes ◊ Tuples en double approximatifs ◊ Tuples en double incohérents ◊ Violation de contrainte métier	
Relations	◊ Hétérogénéité des syntaxes	◊ Violation de l'intégrité référentielle ◊ Référence incorrecte ◊ Circularité entre les tuples ◊ Violation de contrainte métier ◊ Tuples en double approximatifs ◊ Tuples en double incohérents	
Enregistrements/Attributs	◊ Violation de syntaxe ◊ Erreurs d'orthographe	◊ Valeur incorrecte ◊ Violation de domaine ◊ Valeur imprécise ◊ Violation de dépendance fonctionnelle ◊ Valeur Ambiguë ◊ Valeur obsolète ◊ Valeurs contradictoires ◊ Mauvais type de données ◊ Entrée fictive	◊ Valeurs manquantes ◊ Tuples semi-vides ◊ Tuples manquants

2.3. Modèles et Dimensions de la qualité de données

2.3.1. Modèles de la qualité de données

La définition des dimensions et leurs paramètres est une tâche indispensable pour évaluer la qualité de données. Toutefois, aucun accord général sur l'ensemble des dimensions définissant la qualité de données ou sur la signification exacte de chaque dimension n'existe [Batini, 09]. Wang et Strong [Wang, 96] définissent une "dimension" comme étant un ensemble d'attributs de qualité de données qui représentent un aspect ou une construction unique de la qualité de données. Il s'agit d'une propriété de qualité de données mesurable qui représente certaines caractéristiques de données.

Le standard ISO/IEC 25012 [ISO-IEC 25012, 06] propose un modèle selon deux points de vue :

- **Qualité interne et externe de données** : la qualité interne de données représente la capacité d'un ensemble d'attributs de données statiques à satisfaire les besoins. Un exemple de caractéristique de qualité interne de données est la *Cohérence*. La qualité externe de données, quant à elle, représente la capacité des données à satisfaire les besoins lorsque les données sont utilisées dans des conditions spécifiées au sein d'un système informatique. Un exemple de caractéristique de qualité externe de données est la *Sécurité*.
- **Qualité de données en usage** : elle représente la capacité des données à permettre à des utilisateurs spécifiques d'atteindre des objectifs spécifiques en termes d'efficacité, de productivité, de sécurité et de satisfaction dans des contextes d'utilisation spécifiques. La qualité des données en usage vise à définir les caractéristiques de qualité qui expriment le point de vue subjectif de l'utilisateur sur les données sur lesquelles il travaille. La qualité des données utilisées est divisée en quatre caractéristiques : *Efficacité, Productivité, Sécurité* et *Satisfaction*.

Le modèle proposé par le standard ISO/IEC 25012 catégorise les attributs de qualité en quinze caractéristiques considérées selon un point de vue "inhérent" ou "dépendant du système" :

- **Qualité de données inhérente** : la qualité de données, d'un point de vue inhérent, se réfère aux données elles-mêmes, en particulier pour les valeurs de domaine et les restrictions possibles, les relations entre les valeurs (par exemple la cohérence) et les métadonnées.
- **Qualité de données dépendante du système** : la qualité de données dépendant du système se réfère à la mesure dans laquelle la qualité de données est atteinte et conservée dans un système informatique. De ce point de vue, la qualité des données dépend du domaine technologique dans lequel les données sont utilisées. Elle est réalisée par les capacités des composants des systèmes informatiques tels que les dispositifs matériels (par exemple, pour rendre les données disponibles ou pour obtenir la précision requise), les logiciels des systèmes informatiques (comme les logiciels de sauvegarde pour assurer la restauration de données) et d'autres logiciels (pour réaliser la portabilité, par exemple).

La Table 2.2 résume les quinze caractéristiques déterminées par la norme ISO/IEC 25012.

Table 2.2. Dimensions de qualité des données [ISO-IEC 25012, 06]

Caractéristiques	Inhérentes	Dépendantes du système
Exactitude (Accuracy)	X	
Exhaustivité (Completeness)	X	
Cohérence (Consistency)	X	
Crédibilité (Credibility)	X	
Actualité (Currentness)	X	
Accessibilité (Accessibility)	X	X
Conformité (Compliance)	X	X
Confidentialité (Confidentiality)	X	X
Efficacité (Efficiency)	X	X
Précision (Precision)	X	X
Traçabilité (Traceability)	X	X
Compréhensibilité (Understandability)	X	X
Disponibilité (Availability)		X
Portabilité (Portability)		X
Récupérabilité (Recoverability)		X

La définition d'une dimension diffère selon le contexte d'utilisation. Dans la section suivante, nous détaillons pour chaque dimension représentée dans la Table 2.2 quelques définitions issues de la littérature.

2.3.2. Dimensions de la qualité de données

2.3.2.1. Exactitude

Pour ISO/IEC 25012, l'Exactitude représente le degré auquel les données ont des attributs qui représentent correctement la valeur réelle des attributs prévus d'un concept ou d'un événement dans un contexte d'utilisation spécifique [ISO-IEC 25012, 06]. Wang et Strong [Wang, 96] définissent l'Exactitude comme étant la mesure dans laquelle les données sont correctes, fiables et certifiées. Pour Becker et al. 2009 [Becker, 09], l'Exactitude fait référence à la précision des données ; c'est le degré de concordance de l'objet dans la base de données avec l'objet du monde réel. L'Exactitude d'une donnée fait référence au degré de proximité de sa valeur v avec une valeur v' dans un domaine d'attribut considéré comme correct par le système. Si la valeur de la donnée v est la même que la valeur correcte v', la donnée est dite exacte [Fox, 94]. Par exemple, si on considère une entité EMPLOYEE (identifiée par le numéro d'employé 314151) et l'attribut Annee_de_Naissance : si la valeur de l'année de naissance de l'employé 314151 correspond à l'année de naissance réelle de cet employé, alors la donnée est exacte. Ballou et Pazer [Ballou, 85] confirment cette définition et précisent que les données sont exactes lorsque les valeurs stockées dans la base de données correspondent à des valeurs réelles.

La norme ISO/IEC 25012 distingue deux aspects principaux de l'Exactitude :

- **Exactitude syntaxique** : l'exactitude syntaxique est définie comme étant la proximité des valeurs des données avec un ensemble de valeurs définies dans un domaine considéré comme étant syntaxiquement correct. Par exemple, un faible degré d'exactitude syntaxique se produit lorsque le prénom *Mary* est enregistré en tant que *Marj* (parce que, syntaxiquement le mot *Marj* n'est pas un prénom).

- **Exactitude sémantique** : l'exactitude sémantique est définie comme étant la proximité des valeurs des données avec un ensemble de valeurs définies dans un domaine considéré comme étant sémantiquement correct. Par exemple, un faible degré d'exactitude sémantique se produit lorsque le prénom *Mary* est enregistré sous le prénom *Marc*. Les deux prénoms sont syntaxiquement exacts, en raison du domaine de référence dans lequel ils résident, mais *Marc* est un prénom différent.

2.3.2.2. Exhaustivité

Dans la littérature, les définitions de l'Exhaustivité diffèrent selon le contexte auquel elles se réfèrent. Par exemple, pour Becker et al. [Becker, 09], l'Exhaustivité signifie que toutes les valeurs nécessaires de l'objet du monde réel doivent être stockées. Par exemple, un nom de famille ou une adresse qui manquent constitueraient une violation de cette dimension. Batini et al. [Batini, 09] définissent l'Exhaustivité comme étant le degré auquel une collection de données décrit l'ensemble d'objets du monde réel. Wand et Wang [Wand, 96] représentent l'Exhaustivité par la capacité d'un système d'information à représenter chaque état significatif d'un système du monde réel. Dans [Pipino, 02], trois types de d'Exhaustivité sont identifiés :

- **Exhaustivité du schéma** : elle représente le degré auquel les entités et les attributs ne manquent pas dans le schéma.
- **Exhaustivité de la colonne** : elle évalue les valeurs manquantes dans une colonne d'une table.
- **Exhaustivité de la population** : elle évalue les valeurs manquantes par rapport à une population de référence. Par exemple, si une colonne doit contenir au moins une occurrence des 50 états des États-Unis d'Amérique mais qu'elle ne contient que 43 états, alors nous avons une incomplétude de population.

2.3.2.3. Cohérence

Pour Rafique et al. [Rafique, 12], la Cohérence représente le degré auquel l'information possède des attributs cohérents et sans contradiction avec d'autres informations dans un contexte d'utilisation spécifique. Selon Bovee et al. [Bovee, 03], la Cohérence exige que les enregistrements multiples des valeurs des attributs d'une entité soient identiques ou similaires dans le temps et l'espace. Cela signifie que les données appartiennent à une structure logique raisonnable [Becker, 09]. Une erreur serait un enfant de cinq ans dont l'état matrimonial est "marié" ou le cas des codes postaux qui ne sont pas dans une plage autorisée. La Cohérence des données indique si la relation logique entre les données corrélées est correcte et complète [Cai, 15]. Dans le domaine des bases de données, cela signifie généralement que les mêmes données situées dans des zones de stockage différentes doivent être considérées comme étant équivalentes. L'équivalence signifie que les données ont la même valeur et la même signification. La synchronisation de données est le processus qui consiste à rendre les données cohérentes.

2.3.2.4. Crédibilité

Pour ISO/IEC 25012, la Crédibilité représente le degré auquel les données satisfont les besoins des utilisateurs [ISO-IEC 25012, 06]. Il s'agit de la mesure dans laquelle l'information est fiable [Rafique, 12]. Elle représente le degré de croyance sur l'information, basé sur son exactitude, son exhaustivité, sa cohérence et son caractère non fictif [Laranjeiro, 15]. Pour Bovee et al. [Bovee, 03], quelque chose de

crédible est définie comme ayant suffisamment de preuves pour être crue. Les preuves de la crédibilité de la source peuvent être évaluées sans examiner les informations elles-mêmes. Selon Cai et Zhu [Cai, 15], la Crédibilité est utilisée pour évaluer des données non-numériques. Elle fait référence aux composants objectifs et subjectifs de la crédibilité d'une source ou d'un message. La Crédibilité des données a trois facteurs clés : la fiabilité des sources de données, la normalisation des données et le moment où les données sont produites.

2.3.2.5. Actualité

Pour Fox et al. [Fox, 94], une donnée est dite actuelle, ou à jour, à l'instant t si elle est correcte à cet instant t. Une donnée est obsolète à l'instant t si elle est incorrecte à t mais était correcte à un moment donné avant t. Ainsi, être à jour est tout simplement être correct à l'heure actuelle, et être obsolète est un cas particulier d'inexactitude - une inexactitude causée par un changement au fil du temps (évolution temporelle). Rafique et al. [Rafique, 12] définissent l'Actualité par le degré auquel les informations peuvent être identifiées comme étant à jour.

2.3.2.6. Accessibilité

Pour ISO/IEC 25012, l'Accessibilité est la capacité d'accès aux données, en particulier pour les personnes qui ont besoin d'une technologie de support ou d'une configuration spéciale [ISO-IEC 25012, 06]. Il s'agit de la vitesse et de la facilité de localisation et d'obtention d'un objet d'information relatif à une activité particulière [Stvilia, 07]. L'Accessibilité est étroitement liée à l'ouverture des données, plus le degré d'ouverture des données est élevé, plus le degré d'accessibilité est élevé [Cai, 15]. Pour Laranjeiro et al. [Laranjeiro, 15], l'Accessibilité fait référence au degré auquel on peut accéder aux données dans un contexte d'utilisation spécifique. Cela comprend également la pertinence de la représentation de données. Strong et al. [Strong, 97] présentent un modèle d'Accessibilité qui détaille les problèmes d'Accessibilité aux données selon trois catégories :

- des problèmes d'accessibilité technique tels que le réseau informatique et l'obtention des droits d'accès aux données.
- des problèmes de représentation de données tels que l'interprétation et la compréhension des données.
- des problèmes de volume de données pouvant ralentir l'accès aux données.

2.3.2.7. Conformité

Selon [Viscusi, 14], la Conformité consiste à garantir que les données sont publiées conformément à la législation et à l'ensemble des normes en vigueur. Par exemple, la mesure de la conformité d'un ensemble de données publié sur un portail public consiste à calculer le degré de respect d'un décret législatif et/ou des normes de transparence et de diffusion en public.

2.3.2.8. Confidentialité

Une donnée est considérée comme étant confidentielle lorsqu'elle ne peut être lue que par les entités autorisées. La Confidentialité représente le degré auquel l'information possède des attributs qui garantissent qu'elle n'est accessible et interprétable que par les utilisateurs autorisés dans un contexte

d'utilisation spécifique [Rafique, 12]. En matière de Qualité Logicielle, la Confidentialité fait référence à la capacité du logiciel de fournir une protection contre la divulgation non-autorisée de données ou d'informations, qu'elle soit accidentelle ou délibérée [ISO-IEC 25010, 08]. La Confidentialité d'un système représente le degré auquel le système peut garantir qu'un utilisateur non-autorisé ne sera pas en mesure de comprendre les informations protégées par le système [ISO-IEC 25010, 08].

2.3.2.9. Efficacité

Selon ISO/IEC 25012, l'Efficacité correspond à la capacité de traitement des données (accès, acquisition, mise à jour, etc.) et à la fourniture des niveaux de performance appropriés dans des conditions déterminées [ISO-IEC 25012, 06]. L'Efficacité correspond souvent à :

- **L'utilisation des ressources** : la capacité de stockage ou d'accès aux données en utilisant les quantités et les types de ressources appropriés dans les conditions indiquées. Par exemple, l'utilisation de plus d'espace que nécessaire pour sauvegarder des données entraîne un gaspillage de stockage, de mémoire et de processeur.
- **Le comportement temporel** : la capacité des données d'être disponibles pour la lecture ou la modification dans un délai optimal.
- **La conformité** : la capacité des données à respecter les normes, les conventions et les réglementations en vigueur ainsi que les règles similaires en matière d'efficacité.

L'Efficacité est différente de l'Effectivité. Cette dernière fait référence à la capacité des données à permettre aux utilisateurs d'atteindre des objectifs spécifiques d'un point de vue quantitatif et qualitatif dans un contexte d'utilisation spécifique [ISO-IEC 25012, 06].

2.3.2.10. Précision

Le standard ISO/IEC 25012 définit la Précision par la capacité de la valeur assignée à un attribut de fournir le degré d'information nécessaire dans un contexte d'utilisation déclaré [ISO-IEC 25012, 06]. Par exemple, pour représenter la durée d'une course de marathon, l'unité de temps doit être en "secondes" alors que pour représenter la durée d'une course de 100 mètres, l'unité de temps doit être en "millisecondes". Fox et al. [Fox, 94] confirment cette définition et lient la Précision au détail de mesure ou de classification utilisé pour spécifier le domaine d'un attribut. Par exemple, on peut considérer que "vingt valeurs" possibles dans un domaine pour l'attribut 'Couleur' peut être plus précis que seulement "trois valeurs". Ainsi, la précision dépend de la structure du domaine et non d'une donnée particulière.

La Précision est parfois confondue avec l'Exactitude. Cette dernière est une mesure de la correction (à quel degré la donnée est correcte), tandis que la Précision est liée au détail de la mesure (quel est le degré de détail de la mesure de la donnée).

2.3.2.11. Traçabilité

La Traçabilité représente le degré auquel la source d'information, y compris le propriétaire et/ou l'auteur de l'information, et toute modification apportée à l'information peuvent être vérifiées [Rafique, 12]. Il s'agit de la mesure dans laquelle les données sont bien documentées, vérifiables et facilement attribuables à une source [Wang, 96].

2.3.2.12. Compréhensibilité

Les données sont qualifiées de compréhensibles quand elles sont claires, sans ambiguïté et faciles à comprendre. Pour Rafique et al. [Rafique, 12], la Compréhensibilité de données fait référence au degré auquel les informations ont des attributs qui permettent de les lire et de les interpréter, et sont exprimées dans des langues, des unités et des symboles appropriés dans un contexte d'utilisation spécifique. Certaines informations sur la compréhensibilité des données sont fournies par les métadonnées [ISO-IEC 25012, 06]. Par exemple, pour représenter un État, l'acronyme standard est plus compréhensible qu'un code numérique.

2.3.2.13. Disponibilité

Pour Rafique et al. [Rafique, 12], la Disponibilité représente la mesure dans laquelle les informations possèdent des attributs permettant leur récupération par des applications et/ou des utilisateurs autorisés dans un contexte d'utilisation spécifique. Cai et Zhu [Cai, 15] définissent la Disponibilité comme étant le degré de commodité dans lequel les utilisateurs peuvent obtenir des données et des informations connexes. Elle comprend deux sous-éléments :

- **Accessibilité** : elle fait référence à la facilité d'accès (interface utilisateur, des web services, accès public ou privé, etc.).
- **Rapidité d'exécution** : elle fait référence à la vitesse d'accès aux données (par exemple, les données doivent arriver à temps dans un délai donné, l'intervalle de temps entre la collecte et le traitement des données doit répondre aux exigences, etc.).

En matière de Qualité Logicielle, la Disponibilité fait référence à la mesure dans laquelle un composant logiciel est opérationnel et disponible en cas d'utilisation requise [ISO-IEC 25012, 06]. Elle peut être évaluée par la proportion du temps total pendant lequel le logiciel est en état de fonctionnement. La Disponibilité est donc une combinaison de la maturité (qui régit la fréquence des défaillances), de la tolérance aux pannes et de la capacité de récupération (qui régit la durée d'immobilisation après chaque défaillance).

2.3.2.14. Portabilité

La Portabilité fait référence au degré auquel l'information possède des attributs permettant de la remplacer ou de la déplacer d'un système vers un autre, tout en préservant la qualité existante [Rafique, 12]. En matière de Qualité Logicielle, la Portabilité représente le degré de facilité avec laquelle un système ou un composant peut être transféré d'un environnement matériel ou logiciel à un autre [ISO-IEC 25010, 08]. Selon ISO/IEC 25012, la Portabilité de données représente la capacité de transférer des données d'un environnement technologique à un autre tout en conservant l'ensemble de données homogène et cohérent au sein de la plateforme de destination [ISO-IEC 25012, 06].

2.3.2.15. Récupérabilité

La Récupérabilité fait référence au degré auquel les informations possèdent des attributs qui leur permettent de maintenir et de préserver un niveau spécifié d'opérations et de qualité dans un contexte d'utilisation spécifique [Rafique, 12]. Selon ISO/IEC 25012, la Récupérabilité représente la capacité des données à maintenir et à préserver leur intégrité physique et logique, même en cas de défaillance [ISO-

IEC 25012, 06]. En matière de Qualité Logicielle, la Récupérabilité représente le degré auquel un logiciel peut rétablir un niveau de performance spécifié et récupérer les données directement affectées en cas de défaillance [ISO-IEC 25010, 08]. La récupération de données peut être assurée par des fonctionnalités telles que les points de validation et de synchronisation, la restauration ou les mécanismes de sauvegarde et de récupération.

2.4. Mesures des dimensions de la qualité de données

Afin de quantifier la qualité de données, il faut commencer par déterminer les dimensions à mesurer selon les problèmes de qualité identifiés ainsi que les considérations de chaque projet. Pour mesurer une dimension, une ou plusieurs métriques peuvent y être associées. Une métrique de qualité définit la manière d'évaluer une dimension. La mesure de qualité peut être "objective" lorsqu'elle est basée sur des métriques "quantitatives" (par exemple le résultat d'une condition ou d'une équation mathématique) ou "subjective" lorsqu'elle est basée sur des évaluations "qualitatives" effectuées par des administrateurs d'informations ou des utilisateurs (par exemple, à travers un questionnaire, des enquêtes auprès des utilisateurs, etc.) [Redman, 13], [Batini, 09]. Les métriques subjectives mesurent les perceptions, les besoins et les expériences des parties prenantes. Les métriques objectives peuvent avoir trois formes fonctionnelles différentes [Cappiello, 04] :

- **Ratio Simple** : le ratio simple mesure le rapport entre les résultats requis et les résultats réels. Le ratio simple est généralement normalisé entre 0 et 1, en supposant que 1 représente le score le plus élevé et 0 le plus bas.
- **Min ou Max** : la forme fonctionnelle minimale ou maximale est appliquée pour gérer les dimensions qui nécessitent l'agrégation de plusieurs indicateurs de qualité de données. Elle est également utilisée pour fournir une valeur agrégée de la qualité de données sur une seule dimension pour un ensemble de données.
- **Moyenne pondérée** : la moyenne pondérée est la moyenne d'un certain nombre de valeurs affectées de coefficients (les poids). Si une organisation comprend bien l'importance de chaque variable dans l'évaluation globale d'une dimension, une moyenne pondérée des variables est appropriée. Pour obtenir un résultat normalisé, les poids doivent être compris entre 0 et 1 et leur sommation doit être évaluée à 1.

D'autres méthodes comme la moyenne, la somme et la distance peuvent être utilisées dans le modèle d'évaluation en fonction des besoins. Dans la suite de cette section, nous donnons des exemples de mesures objectives pour deux dimensions : l'exactitude et l'exhaustivité.

2.4.1. Mesure de la dimension "Exactitude"

L'Exactitude fait référence à la distance entre une valeur v et une autre valeur v' considérée par le système comme correcte. Deux types d'exactitude peuvent être identifiés : l'exactitude syntaxique et l'exactitude sémantique. L'exactitude syntaxique est mesurée au moyen de fonctions de comparaison qui évaluent la distance entre v et v'. La distance d'édition entre les deux valeurs peut par exemple être calculée grâce à la distance de Levenshtein qui correspond au nombre minimal de caractères qu'il faut

supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre. Si on reprend l'exemple des prénoms *Mary* et *Marj*, la distance de Levenshtein est égale à 1.

L'exactitude sémantique capture les cas dans lesquels v est une valeur syntaxiquement correcte (par exemple *Marc*), mais elle est différente de v' (par exemple *Mary*). Il est souvent possible de détecter un problème d'exactitude sémantique dans un enregistrement et de fournir une valeur correcte en comparant l'enregistrement avec des données équivalentes provenant de sources différentes. Cela nécessite la capacité de reconnaître que deux enregistrements font référence à la même entité du monde réel. Cette opération est souvent réalisée par des algorithmes de "couplage d'enregistrements" que nous verrons un peu plus loin dans ce chapitre. Nous avons proposé, dans [Talha, 20-b], plusieurs techniques et différents scénarios pour évaluer l'exactitude de données dans un contexte Big Data.

2.4.2. Mesure de la dimension "Exhaustivité"

L'Exhaustivité correspond au degré de "complétude" auquel les données décrivent une entité du monde réel. Dans un modèle relationnel, l'exhaustivité peut être valorisée en fonction de la présence et de la signification des valeurs NULL. Généralement, une valeur NULL signifie que la valeur peut être manquante et, si c'est le cas, il est nécessaire de comprendre pourquoi cette valeur est manquante. En effet, une valeur peut être manquante car elle existe mais n'est pas connue ou parce qu'elle n'existe pas ou parce que son existence est inconnue. Considérons l'exemple de la Table 2.3 qui représente une relation *Personne*, avec les attributs *Prénom*, *Nom*, *Date_Naissance* et *Email*.

Table 2.3. Relation « Personne » non-exhaustive

ID	Prénom	Nom	Date_Naissance	Email
1	John	Doe	01/03/1983	NULL
2	Edward	Smith	06/02/1982	e.smith@mycomp.org
3	Anthony	Blair	14/12/1975	NULL
4	Marianne	White	13/02/1992	NULL

Email réellement n'existe pas

Email existe mais inconnu

On ne sait pas si l'Email existe

Dans cet exemple, la valeur de l'email est manquante pour les tuples ayant les identifiants 1, 3 et 4. Si la personne représentée par le tuple 1 n'a réellement pas d'adresse email alors il n'y a pas d'incomplétude ; si la personne représentée par le tuple 3 possède bien une adresse email mais on ne connait pas sa valeur, alors le tuple 3 est incomplet ; enfin, si on ne sait pas si la personne représentée par le tuple 4 possède une adresse email ou pas, alors dans ce cas, on ne peut pas déterminer si le tuple est complet ou pas. Pour mesurer l'Exhaustivité de l'attribut *Email* de la relation *Personne* de cet exemple, on se réfère à l'administrateur des informations : s'il décide que le cas du tuple 4 peut être considéré comme incomplet (l'email est obligatoire), alors l'exhaustivité de l'attribut Email est égale à 50%, sinon, s'il considère qu'il est complet (l'email est facultatif), alors l'exhaustivité vaut 75%.

2.5. Systèmes de gestion de la qualité de données

Un système de gestion de la qualité de données consiste en deux phases principales :

- **Evaluation de la qualité de données** : elle consiste à identifier et à quantifier les problèmes de qualité dans un système d'information.
- **Amélioration de la qualité de données** : elle consiste à corriger les problèmes identifiés dans la phase précédente.

2.5.1. Evaluation de la qualité de données

L'évaluation de la qualité de données consiste à mesurer certaines dimensions de qualité pertinentes déterminées en fonction des problèmes de qualité. Batini et al. [Batini, 09] listent cinq étapes pour la phase d'évaluation de la qualité de données :

1. **Analyse de données** : cette étape consiste à examiner les schémas des données et à effectuer des entretiens pour bien comprendre les données et les règles d'architecture et de gestion associées.
2. **Analyse des exigences en matière de qualité de données** : cette étape consiste à examiner l'opinion des utilisateurs et des administrateurs pour identifier les problèmes de qualité et définir de nouveaux objectifs de qualité.
3. **Identification des zones critiques** : cette étape consiste à identifier les bases de données et les flux de données les plus pertinents à évaluer quantitativement.
4. **Modélisation des processus** : cette étape consiste à fournir un modèle des processus produisant ou mettant à jour les données.
5. **Mesure de la qualité** : cette étape consiste à, dans un premier temps, sélectionner les dimensions de qualité affectées par les problèmes de qualité identifiés lors de l'étape d'analyse des exigences de qualité et, ensuite, définir les métriques correspondantes (à chaque dimension on peut identifier une ou plusieurs métriques).

2.5.2. Amélioration de la qualité de données

La phase d'amélioration de la qualité de données s'intéresse à l'identification des étapes, des stratégies et des techniques pour atteindre les objectifs de qualité souhaités. Dans [Batini, 09], les auteurs listent 10 étapes d'amélioration de la qualité de données :

1. **Evaluation des coûts** : estimer les coûts directs et indirects de la qualité de données.
2. **Attribution des responsabilités de processus** : identifier les propriétaires des processus et définir leurs responsabilités en matière de production de données et d'activités de gestion.
3. **Attribution des responsabilités en matière de données** : identifier les propriétaires des données et définir leurs responsabilités en matière de gestion de données.
4. **Identification des causes des erreurs** : identifier les sources des problèmes de qualité.
5. **Sélection des stratégies et des techniques** : identifier toutes les stratégies d'amélioration des données et les techniques correspondantes, conformément aux connaissances contextuelles, aux objectifs de qualité et aux contraintes budgétaires.
6. **Conception de solutions d'amélioration des données** : choisir la stratégie la plus efficace et la plus effective ainsi que les techniques et les outils associés, pour améliorer la qualité de données.

7. **Contrôle des processus** : définir des points de contrôle dans les processus de production de données, afin de contrôler la qualité pendant leurs exécutions.
8. **Refonte des processus** : définir les actions d'amélioration des processus pouvant apporter des améliorations à la qualité des données correspondantes.
9. **Gestion de l'amélioration** : définir de nouvelles règles d'organisation pour la qualité de données.
10. **Surveillance de l'amélioration** : établir des activités de surveillance périodiques qui fournissent un retour sur les résultats des processus d'amélioration et permettent leurs réglages dynamiques.

On distingue deux types généraux de stratégies d'amélioration, à savoir : les stratégies d'amélioration des données et les stratégies d'amélioration des processus.

2.5.2.1. Stratégies d'amélioration de données

Ces stratégies consistent à améliorer la qualité des données en modifiant directement leurs valeurs. Par exemple, la suppression des doublons, l'actualisation des valeurs obsolètes (remplacer des valeurs périmées par des valeurs plus récentes), la correction des valeurs incorrectes, etc. Plusieurs techniques peuvent être utilisées pour améliorer les données. Parmi ces techniques, nous pouvons citer :

- L'acquisition de nouvelles données pour remplacer celles qui posent des problèmes de qualité.
- La normalisation des valeurs en remplaçant ou en complétant les valeurs de données non standard par des valeurs correspondantes conformes à la norme. Cela consiste à remplacer les pseudonymes par des noms correspondants (par exemple, remplacer momo par Mohamed), les abréviations par les noms complets correspondants (par exemple, remplacer Blvd. Mohamed VI par Boulevard Mohamed VI), etc.
- Le couplage d'enregistrements. Cette technique consiste à identifier les représentations de données dans différents ensembles de données qui peuvent faire référence à un même objet.
- L'intégration de données et de schémas. Ces processus consistent à définir une vue unifiée des données fournies par des sources de données hétérogènes ce qui permet aux utilisateurs d'accéder aux données issues de sources hétérogènes via une vue unifiée.

2.5.2.2. Stratégies d'amélioration de processus

Ces stratégies consistent à améliorer la qualité de données en redéfinissant les processus qui créent ou modifient des données. Par exemple, ajouter une étape de contrôle des formats de données avant le stockage, ajouter une étape de validation de la fiabilité d'une source de données lors de la collecte de données, etc. Les principales techniques utilisées pour améliorer les processus sont :

- La mise en œuvre des procédures de contrôle des processus de production de données lors de leur création ou de leur mise à jour. L'intérêt de cette procédure est d'éviter la dégradation de la qualité de données et de la propagation des erreurs.
- La refonte de tous les processus de production des données afin d'éliminer les causes de la mauvaise qualité.

2.6. Techniques d'amélioration de la qualité de données en Big Data

2.6.1. Nettoyage de données

Pour être traitées et interprétées de manière efficace, les données doivent satisfaire à un ensemble de caractéristiques. Les données satisfaisant ces caractéristiques sont qualifiées de "bonne" qualité sinon elles sont considérées comme étant des données "sales". Les données sales conduisent à des résultats erronés et à des statistiques trompeuses [Rahm, 00]. Parmi les techniques d'amélioration de la qualité de données sales on trouve le nettoyage de données (Data Cleansing, en anglais) qui doit être une condition préalable à toute tâche de traitement de données [Gschwandtner, 12]. Il s'agit du processus de détection et de correction des données corrompues (par exemples, des données en double, des données manquantes, des données incohérentes, des données erronées, etc.).

L'évaluation de la qualité de données peut être utilisée pour quantifier la nécessité du nettoyage de données. Ce processus de nettoyage est appliqué en particulier lorsqu'un entrepôt de données est construit en collectant des données depuis différentes sources ou lorsque plusieurs bases de données sont fusionnées. Ainsi, les enregistrements en double apparaîtront dans la base de données fusionnée et les erreurs dans les bases sources seront transmises dans les bases finales. Le problème consiste donc à identifier et à éliminer les erreurs et les incohérences dans les données.

Le nettoyage de données est un processus qui consiste à appliquer un ensemble d'opérations aux données existantes pour supprimer les anomalies et obtenir un ensemble de données constituant une représentation précise et unique. Il peut être appliqué lors de l'acquisition des nouvelles données ou de l'amélioration de la qualité des données existantes dans un système d'information [Maletic, 00]. Pour Raman et Hellerstein [Raman, 00], le processus de nettoyage de données comporte trois composants : l'audit des données pour détecter les écarts, le choix des transformations pour les corriger, et leur application sur l'ensemble de données.

Il s'agit d'un processus automatique, ou semi-automatique, et qui s'exécutent, selon [Müller, 03], dans l'ordre suivant :

1. Adapter le format pour les tuples et les valeurs.
2. Appliquer les contraintes d'intégrité.
3. Dériver les valeurs manquantes des valeurs existantes.
4. Eliminer les contradictions à l'intérieur ou entre les tuples.
5. Fusionner et éliminer les doublons.
6. Détecter les valeurs aberrantes (les tuples et les valeurs ayant un potentiel élevé d'invalidité).

Nous déduisons donc qu'un processus de nettoyage de données est composé principalement de trois phases :

- **Phase d'audit** : cette première phase consiste à auditer les données pour définir et déterminer les types d'anomalies qu'elles contiennent.
- **Phase de spécification** : cette phase consiste à identifier les instances d'anomalies dans les ensembles de données.
- **Phase de correction** : cette dernière phase consiste à exécuter des techniques pour éliminer toutes les anomalies identifiées à l'étape précédente.

Chacune de ces phases constitue un problème complexe en soi. Différentes méthodes et techniques peuvent être appliquées dans un processus de nettoyage de données. En voici quelques-unes :

- **Analyse statistique** : cette technique consiste à analyser les données en utilisant les valeurs de la moyenne, de l'écart type, des algorithmes de regroupement, etc. Cela permet d'identifier des valeurs inattendues indiquant des tuples invalides possibles. Ces méthodes statistiques peuvent être utilisées pour l'audit de données ainsi que pour la correction d'anomalies [Maletic, 00].
- **Analyse syntaxique** : cette technique est effectuée pour détecter les erreurs de syntaxe. Un analyseur pour une grammaire G est un programme qui décide pour une chaîne donnée si elle appartient au langage défini par la grammaire G. Dans le nettoyage de données, les chaînes sont des tuples complets d'une instance relationnelle ou des valeurs d'attributs provenant d'un domaine. Les chaînes qui représentent des erreurs de syntaxe doivent être corrigées. Cela peut être fait par exemple en utilisant les fonctions d'édition de distance en choisissant la correction possible avec la distance d'édition minimale [Müller, 03].
- **Déduplication de données** : cette technique permet de contrôler la granularité des données en réduisant les données redondantes. Elle permet ainsi d'assurer la copie unique des données stockées via la détection et la suppression des données en double.
- **Application des contraintes d'intégrité** : cette méthode garantit la satisfaction des contraintes d'intégrité après l'exécution des opérations qui modifient les jeux de données en insérant, en supprimant ou en mettant à jour des tuples [Mayol, 99]. Les approches les plus connues sont la vérification et la maintenance des contraintes d'intégrité :
    - La vérification des contraintes d'intégrité rejette les opérations qui, si elles étaient appliquées, violeraient une contrainte d'intégrité.
    - La maintenance des contraintes d'intégrité concerne l'identification des mises à jour supplémentaires à ajouter à l'opération d'origine pour garantir que la collecte de données résultante ne viole aucune contrainte d'intégrité.
- **Transformation de données** : cette technique consiste à mapper les données provenant de diverses sources dans un schéma commun qui correspond aux besoins de l'application envisagée. La correction des valeurs doit être effectuée uniquement dans les cas où les données d'entrée ne sont pas conformes au schéma cible [Müller, 03].
- **Détection des erreurs (à base de modèles)** : cette méthode consiste à identifier les champs et les enregistrements aberrants qui ne sont pas conformes aux modèles existants dans les données. Des techniques combinées (partitionnement, classification et regroupement) permettent d'identifier les modèles qui s'appliquent à la plupart des enregistrements. Un modèle est défini par groupe d'enregistrements ayant des caractéristiques similaires pour p% des champs du jeu de données, où p est une valeur définie par l'utilisateur (généralement supérieure à 90) [Maletic, 00].

Le nettoyage de données est une approche d'amélioration de la qualité de données mais reste insuffisante. Elle ne permet pas notamment de régler des anomalies comme les valeurs manquantes, les tuples semi-vides ou les tuples manquants. Ces anomalies peuvent être corrigées moyennant d'autres techniques telles que l'intégration de données. Par ailleurs, la majorité des techniques de nettoyage de données se basent sur des données structurées (relationnelles) ce qui n'est généralement pas le cas en Big Data. Les données non-structurées nécessitent de l'analyse sémantique pour comprendre les relations entre les données et pouvoir ainsi identifier et nettoyer les lacunes. En outre, la grande volumétrie de données

représente un autre défi pour ces techniques de nettoyage qui nécessitent le chargement en mémoire de plusieurs ensembles de données.

2.6.2. Intégration de données

L'intégration de données consiste en un ensemble de processus utilisés pour récupérer et combiner des données provenant de différentes sources. Les techniques d'intégration de données traditionnelles étaient principalement basées sur le processus ETL (Extract, Transform et Load) pour ingérer et charger des données dans un entrepôt de données. Les outils ETL combinent trois fonctions principales pour obtenir des données d'un environnement source et les placer dans un autre environnement cible :

- **Extraire** (Extract) : lire les données de la base de données source.
- **Transformer** (Transform) : convertir le format des données extraites afin qu'il soit conforme aux exigences de la base de données cible.
- **Charger** (Load) : écrire les données dans la base de données cible.

Les entrepôts de données offrent aux utilisateurs des moyens pour consolider les informations provenant de sources externes afin d'analyser et de générer des rapports sur la qualité de données. Les outils ETL sont utilisés pour transformer les données au format requis par l'entrepôt de données. La transformation est effectuée dans un emplacement intermédiaire avant que les données ne soient chargées dans l'entrepôt de données cible. Traditionnellement, un processus ETL est utilisé dans des traitements par lots sur des données "au repos" ce qui pose un vrai défi pour les flux de données. En effet, dans les environnements Big Data, un volume énorme de données, souvent non-structuré, est collecté à partir de nombreuses sources de données hétérogènes qui génèrent des données en temps réel avec différents niveaux de qualité. Les techniques d'intégration de données traditionnelles ne sont donc pas applicables dans de telles circonstances. Les composants de l'écosystème Big Data, notamment Hadoop et les bases de données NoSQL, chacun d'eux a sa propre approche pour extraire, transformer et charger les données.

Pour intégrer les données en Big Data et assurer ainsi une meilleure qualité des données, trois techniques de base sont utilisées [Dong, 13] : la "Correspondance de Schémas", le "Couplage d'enregistrements" et la "Fusion de Données". Ces techniques ont été utilisées dans l'intégration traditionnelle des données (dans les processus ETL) mais leur application dans un environnement Big Data est bien plus complexe. Dans les sections suivantes, nous allons décrire chacune de ces trois techniques et discuterons les dernières études réalisées pour les appliquer dans des processus d'intégration de données en Big Data.

2.6.3. Correspondance de Schémas

La Correspondance de Schémas (Schema Matching, en anglais) est un problème de recherche critique ouvert depuis les années 1980s et dont la criticité est devenue encore plus complexe avec l'émergence de la Big Data. De nombreux travaux sur la correspondance de schémas existent dans la littérature [Rahm, 01], [Shvaiko, 05], [Bellahsene, 11], [Shvaiko, 13]. Jusqu'au début des années 2000, la correspondance de schémas se faisait manuellement [Do, 02]. Aujourd'hui, de nombreux modèles et approches ont été publiés à ce sujet mais restent malheureusement insuffisants pour obtenir un système de

mise en correspondance de schémas automatique et satisfaisant [Sutanta, 16] ; l'intervention humaine pour réviser et reformuler les résultats de correspondance de schémas reste toujours nécessaire.

La correspondance de schémas est un processus composé de deux phases : la création d'un schéma médiatisé (global) pouvant couvrir toutes les données et, ensuite, l'identification des correspondances entre le schéma médiatisé et le schéma des données sources pour déterminer quels ensembles d'attributs contiennent les mêmes informations [Dong, 15].

Bernstein et al. [Bernstein, 11] définissent un schéma comme étant une structure formelle qui représente un artefact conçu, tel qu'un schéma SQL, un schéma XML, un diagramme entité-relation, une description d'ontologie, une définition d'interface ou une définition de formulaire. Ils définissent également une correspondance comme une relation entre un ou plusieurs éléments d'un schéma et un ou plusieurs éléments d'un autre schéma. Dans les bases de données relationnelles, la correspondance de schémas consiste à lier les tables et les colonnes d'une base de données à celles qui représentent les mêmes concepts dans une autre base de données. Les auteurs de [Bernstein, 11] et [Rahm, 01] exposent une taxonomie de techniques et de méthodes utilisées pour réaliser la mise en correspondance de schémas, parmi lesquelles on trouve :

- La correspondance linguistique basée sur le nom ou la description d'un élément, à l'aide de techniques de radicalisation, de tokenisation, de correspondance de chaînes et de sous-chaînes et de recherche d'informations.
- La correspondance basée sur des informations auxiliaires telles que des thésaurus (i.e. de ensembles de mots standards), des acronymes, des dictionnaires et des listes de discordances.
- La correspondance basée sur les instances qui considère que les éléments de deux schémas sont similaires si leurs instances sont similaires sur la base de statistiques, de métadonnées ou de classificateurs entraînés.
- La correspondance basée sur la structure qui considère les éléments de deux schémas comme étant similaires s'ils apparaissent dans des groupes de structures similaires, ont des relations similaires ou ont des relations avec des éléments similaires.

Pour gérer l'ambiguïté de la signification des attributs, l'ajout de la "probabilité" à la correspondance des attributs et des schémas a été proposé par Das Sarma et al. [Das Sarma, 08]. Cette approche a été évaluée sur des "tables web" explorées à partir de cinq domaines ; chaque domaine contient environ 50 à 800 tables web. Les tables web sont des données tabulaires hétérogènes stockées en HTML et n'ont pas de schéma clair et précis. Une méthode de recherche par mot-clé sur des tables web a été proposée par Cafarella et al. [Cafarella, 08] basée sur deux approches de classement "Feature Rank" et "Schema Rank". Das Sarma et al. [Das Sarma, 12] ont proposé un cadre pour récupérer les tables web qui sont liées à une table donnée. Ils ont expérimenté leur approche sur des tables Wikipédia et ont montré de bons résultats.

2.6.4. Couplage d'enregistrements

Le couplage d'enregistrements est le processus permettant d'identifier les enregistrements qui font référence à la même entité logique dans différents ensembles de données, en particulier en l'absence d'identifiants communs. Ce processus consiste à rassembler des informations provenant de différents enregistrements qu'on suppose liés à une même entité. Cela implique la liaison des enregistrements au

sein d'un même fichier ou entre deux ou plusieurs fichiers pour identifier les enregistrements similaires. Le défi consiste à rassembler les enregistrements des mêmes entités individuelles en cherchant les correspondances exactes [Herzog, 07]. Le couplage d'enregistrements peut être utilisé lors de l'évaluation de la qualité de données (pour détecter les similarités entre les données) ou lors de l'amélioration de la qualité de données (pour nettoyer, intégrer ou fusionner des données). Le but du couplage d'enregistrements est de déterminer l'état de correspondance réelle de chaque paire d'enregistrements : *Correspondance* ou *Non-correspondance*. En Big Data, les données sont hétérogènes dans leurs structures et sont collectées à partir de nombreuses sources qui peuvent être dynamiques et en constante évolution [Dong, 15]. Cela rend le couplage d'enregistrements difficile à réaliser.

Deux principaux types de couplage d'enregistrements existent :

- **Le couplage déterministe** : cette méthode de couplage est relativement simple. Elle requiert généralement l'accord exact sur une clé de correspondance qui peut être un identifiant unique (par exemple, le numéro d'une carte d'identité nationale ou le numéro de l'assurance maladie) ou une collection d'identificateurs partiels (par exemple, une clé composée du nom, du prénom, de l'année de naissance et du code postal de naissance). Une paire d'enregistrements est considérée comme un "lien" uniquement si les clés de correspondance (identifiant unique ou collection d'identificateurs partiels) sont identiques. Malheureusement, comme des erreurs ou un manque d'informations dans les enregistrements peuvent exister, le couplage déterministe n'est pas toujours évident.
- **Le couplage probabiliste** : pour surmonter les limites du couplage déterministe, des méthodes probabilistes ont été proposées pour déterminer le couplage en présence des erreurs dans les enregistrements et/ou sans utiliser les clés de correspondance. Newcombe et al. [Newcombe, 59] ont été les premiers à proposer des méthodes probabilistes, suggérant qu'un poids de correspondance puisse être créé pour représenter la probabilité que deux enregistrements correspondent réellement, compte tenu de l'accord ou du désaccord sur un ensemble d'identificateurs partiels. Fellegi et Sunter [Fellegi, 69] ont ensuite formalisé mathématiquement le couplage d'enregistrements probabiliste sur la base des propositions de Newcombe et al. Une autre forme de couplage d'enregistrements probabiliste, basée sur des modèles statistiques pour mesurer la preuve que les enregistrements représentent les mêmes objets, a été proposée par Copas et Hilton [Copas, 90].

Dans le couplage d'enregistrements, de nombreuses techniques sont utilisées :

- **Appariement par paire** : cette technique est utilisée pour comparer une paire d'enregistrements pour vérifier s'ils appartiennent à la même entité logique.
- **Clustering** : cette technique est utilisée pour parvenir à une décision cohérente du partitionnement des enregistrements afin de s'assurer que chaque partition appartient à une entité distincte.
- **Blocage** : cette technique est utilisée pour partitionner les enregistrements d'entrée en plusieurs blocs pour n'autoriser la correspondance par paires qu'avec les enregistrements d'un même bloc.

Dans des environnements Big Data, Ortez et da Silva [Cortez, 13] ont proposé d'utiliser des techniques d'extraction d'informations sur des données non-structurées pour obtenir des données structurées avant d'utiliser des techniques de couplage traditionnelles. Kannan et al. [Kannan, 11] ont proposé une autre approche pour lier des extraits de texte non-structurés à des données structurées. Leur approche est principalement basée sur : l'analyse sémantique d'un extrait de texte (à l'aide de techniques

de marquage, d'analyse plausible et d'analyse optimale) et une fonction d'appariement qui renvoie un score probabiliste d'appariement entre les enregistrements.

2.6.5. Fusion de données

La fusion de données est une combinaison de techniques qui vise à résoudre les conflits entre plusieurs ensembles de données. C'est un domaine qui a émergé avec la Big Data dans le but d'assurer la véracité de données. Généralement, trois techniques sont appliquées dans la fusion de données :

- **Détection de copie** : cette technique consiste à détecter les sources du copieur de données et réduire leur poids lors de la fusion de données (le fournisseur de l'information a plus de poids que le copieur).
- **Vote** : cette technique consiste à identifier la valeur la plus courante pour chaque attribut.
- **Qualité de la source** : après le vote, accorder plus de poids aux sources d'information ayant le plus grand nombre d'attributs en commun.

De nombreux travaux ont été réalisés pour mesurer la fiabilité des sources de données. Dong et al. [Dong, 09] ont proposé un processus de fusion de données composé de 3 phases : découverte de la vérité, évaluation de la fiabilité et détection des copies. De nombreux chercheurs ont ensuite proposé des extensions pour chaque phase [Galland, 10], [Pasternack, 10], [Yin, 08].

Dans certains domaines, les données en ligne changent au fil du temps et dans de nombreux cas, elles doivent être évaluées en temps réel. Pour résoudre ce problème, Liu et al. [Liu, 11] ont proposé une technique de fusion de données en ligne dans laquelle la précision de la source et la relation de copie sont évaluées en mode hors ligne, et au moment de la réponse à la requête, la découverte de la vérité est évaluée.

2.7. Défis de la qualité de données en Big Data

La Big Data est étroitement liée à la nouvelle (et ancienne) question de « la qualité de données » [Hoeren, 18]. La recherche en qualité de données a abouti à la définition d'un ensemble de dimensions, de métriques et de méthodes pour évaluer et améliorer la qualité des bases de données relationnelles. Ces méthodes, dites classiques, considèrent la qualité de données comme un concept indépendant du contexte dans lequel les données sont produites et utilisées [Arolfoand, 18]. De nombreuses études récentes comme [Ge, 18], [Taleb, 18], [Hoeren, 18], [Arolfoand, 18], [Batini, 15] et [Firmani, 15] ont étudié les particularités de la qualité de données dans le contexte Big Data. Chacune de ces études aborde certains aspects de la Big Data, résumés dans le modèle 5V, pour étudier l'impact sur les dimensions et les métriques de la qualité de données. Il a été reconnu que les méthodes classiques utilisées pour gérer la qualité de données dans les bases de données relationnelles doivent être adaptées et étendues pour capturer les nouvelles caractéristiques de la Big Data [Merino, 15]. Aujourd'hui, aucun cadre standard de gestion de la qualité en Big Data n'existe ; la plupart des travaux existants sont encore en cours d'investigation et n'ont pas atteint un bon niveau de maturité [Taleb, 18].

Parmi les principaux défis discutés dans la littérature pour mettre en place un système de gestion de la qualité de données on trouve :

1. **La volumétrie** : le volume de données est énorme et il est difficile d'évaluer et d'améliorer la qualité de données dans un délai raisonnable [Cai, 15]. Les plateformes Big Data doivent se concentrer essentiellement sur le stockage de données de bonne qualité plutôt que sur de très grandes quantités données non pertinentes, de sorte que de meilleurs résultats et conclusions puissent en être tirés. Cela conduit à diverses questions telles que la manière dont on peut s'assurer que les données sont pertinentes, combien de données seraient suffisantes pour la prise de décision et si les données stockées sont exactes ou non, etc. [Katal, 13].
2. **L'hétérogénéité** : la littérature distingue entre trois types de données : les données structurées, les données semi-structurées et les données non-structurées. Ce troisième type de données occupe la grande partie des données produites aujourd'hui (certaines études estiment que plus de 80% des données qui existent aujourd'hui sont des données non-structurées). Les données structurées sont organisées de façon à simplifier leur intégration. Par contre, les données non-structurées sont très complexes à traiter. En effet, la compréhension de la sémantique et la corrélation entre les données non-structurées est une tâche ardue [Saha, 14] d'autant plus que la conversion de toutes ces données non-structurées en données structurées est impossible [Katal, 13]. En outre, l'hétérogénéité de données, quelles que soient leurs structures, pose des problèmes de représentation des objets (un même objet peut avoir différentes représentations et différents formats) ce qui mène systématiquement à des problèmes de compréhension de données.
3. **La mise à l'échelle (scalabilité)** : l'extension vers des ensembles de données plus importants est au cœur des nouvelles technologies de la Big Data. Les ensembles de données s'étendent en continu et deviennent de plus en plus complexes. De ce fait, les techniques et les processus doivent constituer de la scalabilité une composante importante de la qualité de données. La non-prise en compte de cette exigence entraînerait un ralentissement, voire un blocage d'exécution des outils d'analyse, de gestion et d'exploration de données.
4. **Le changement de données** : aujourd'hui, les données changent très rapidement et peuvent devenir vite obsolètes. Si les organisations ne peuvent pas collecter les données requises, à jour et à temps, elles risquent de produire des conclusions inutiles ou trompeuses, conduisant éventuellement à des erreurs de décision [Cai, 15]. Un des principaux objectifs de la qualité de données est la correction des données erronées, notamment celles qui sont périmées. Ainsi, un système de gestion de la qualité de données doit surveiller en continu leur actualité.
5. **La sécurité de données** : pour que les données soient évaluées et corrigées, l'accès de manière simple à l'ensemble des données est exigé. La sécurité de données peut rendre le processus de gestion de la qualité de données plus lent et plus complexe. La confidentialité des données, à titre d'exemple, peut composer une barrière empêchant ou ralentissant l'accès aux données [Strong, 97]. La mise en place d'un système de gestion de la qualité de données nécessite la mise en œuvre, d'une part, d'un processus d'évaluation de la qualité de données et, d'autre part, d'un processus d'amélioration de la qualité de données. L'évaluation de la qualité de données peut nécessiter l'accès en "Lecture" à plusieurs ensembles de données pour réaliser des tâches de comparaison et de recherche d'informations. L'amélioration de la qualité de données, quant à elle, peut nécessiter des accès en "Ecriture" pour réaliser certaines opérations de nettoyage, de fusion ou de mise à jour de données. Ces deux processus d'évaluation et d'amélioration de la qualité de données peuvent donc être bloqués par les mécanismes de sécurité de données, et il faudra, par conséquent, trouver un compromis entre ces deux systèmes de gestion de la qualité de données et de sécurité de données [Talha, 19].

6. **L'intégration et la fusion de données** : l'intégration de données est un domaine de recherche qui assure l'agrégation de données. Ce processus d'agrégation nécessite que les données soient homogènes et isomorphes alors qu'en Big Data les données et leurs structures sont généralement hétérogènes. La fusion de données présente un moyen efficace d'intégrer, de gérer, de manipuler et de stocker des données volumineuses [Esteban, 05]. En Big Data, les données proviennent de différentes sources dans différents domaines et se composent de plusieurs modalités, chacune ayant une représentation, une distribution, une échelle et une densité différentes [Zheng, 15]. Par conséquent, la fusion efficace des données hétérogènes constitue un grand problème à résoudre dans tout projet Big Data ; il s'agit d'une condition préalable à l'assurance qualité et à l'exploration analytique des données intégrées [Zhang, 18-a].

2.8. Conclusion

L'émergence de la Big Data comme une solution puissante pour exploiter, à une grande vitesse, de très gros volumes de données, souvent hétérogènes, a motivé des grandes (et petites) organisations commerciales et gouvernementales à l'adopter dans de nombreux projets. Toutefois, une mauvaise qualité de données entraînera une faible efficacité d'exploitation de la Big Data et entraînera même de graves erreurs de prise de décision. La qualité de données doit être définie dans un contexte métier et se caractériser par des métadonnées, des règles techniques et métier ainsi que des assertions. Les approches traditionnelles de la qualité de données évoluent rapidement pour répondre aux contraintes de volumétrie et d'hétérogénéité imposées par le contexte Big Data mais il faudra encore beaucoup de temps avant que ces approches arrivent à maturité et aboutissent à terme à une normalisation.

Dans ce chapitre, nous avons présenté la qualité de données, sa définition, certains de ses problèmes et sources de problèmes ainsi que certaines de ses dimensions. Nous avons ensuite discuté les étapes et les techniques de mise en place d'un système de gestion de la qualité de données et avons listé les principaux défis de recherche liés à l'évaluation et à l'amélioration de la qualité de données dans un contexte Big Data. Dans le prochain chapitre, nous enchaînons notre état de l'art sur l'étude détaillée de la sécurité de données en Big Data.

Chapitre 3. Sécurité de données en Big Data

3.1. Introduction

Une des principales caractéristiques de la Big Data est la distribution du stockage et du traitement de données. Selon Sherman [Sherman, 92], sécuriser un système distribué revient essentiellement à :

- assurer la disponibilité des informations communiquées entre, ou détenues dans, les systèmes participants,
- maintenir l'intégrité des informations communiquées entre les systèmes participants ou détenues par ceux-ci. Cela consiste à empêcher la perte ou la modification d'informations en raison d'un accès non-autorisé, d'une défaillance d'un composant ou d'autres erreurs,
- maintenir la confidentialité et le bon fonctionnement des services fournis,
- assurer le contrôle de l'accès aux services ou à leurs composants pour garantir que les utilisateurs ne peuvent utiliser que les services pour lesquels ils sont autorisés,
- authentifier l'identité des partenaires communicants et, en cas de nécessité, assurer la non-répudiation de l'origine et de la livraison des données, et
- le cas échéant, assurer un interfonctionnement sécurisé avec le monde des systèmes non ouverts (par exemple, les applications basées sur le protocole TCP/IP).

La sécurité de données se concentre généralement sur trois objectifs : la confidentialité, l'intégrité et la disponibilité. La confidentialité consiste à protéger les données contre les accès non-autorisés. L'intégrité vise la protection des données contre les modifications non-autorisées. La disponibilité assure l'accessibilité des données aux utilisateurs autorisés. ISO/IEC 27001 considère d'autres propriétés, telles que l'authenticité, la responsabilité, la non-répudiation et la fiabilité, qui peuvent être impliquées dans les systèmes de sécurité [ISO-IEC 27001, 05]. Par ailleurs, et en particulier pour les domaines fondés sur les technologies Big Data tels que les réseaux sociaux, le Cloud Computing, l'IoT et les environnements intelligents, on considère souvent la protection de la vie privée (Privacy en anglais) comme étant un objectif majeur auquel on prête une attention particulière. La protection de la vie privée et la sécurité ont toujours été deux domaines de préoccupation distincts. Pourtant, elles sont de plus en plus discutées ensemble car la sécurité est nécessaire pour protéger la vie privée des individus. Les mécanismes de protection de la vie privée s'appuient sur les techniques de confidentialité et prennent en considération d'autres aspects tels que le consentement des personnes concernées, la conformité aux contraintes réglementaires et légales, etc.

La réalisation de ces objectifs de sécurité dans le contexte Big Data est souvent confrontée à de nombreux défis imposés par le besoin d'architectures distribuées et évolutives, l'externalisation des infrastructures, l'hétérogénéité des environnements, le partage des ressources avec des systèmes étrangers, etc. Relever ces défis serait une tâche difficile dans un environnement volumineux. En effet, les technologies et les méthodes antérieures ne sont plus appropriées et manquent de performances lorsqu'elles sont appliquées dans un contexte Big Data [Dissanayake, 21]. Les techniques de sécurité traditionnelles, quant à elles, ne peuvent pas répondre pleinement aux exigences des applications Big Data [Bao, 18]. Malgré cela, à ce jour, les entreprises utilisent toujours des méthodes classiques pour protéger les données sensibles de leurs clients. Un autre défi de sécurité pour la gestion et l'analyse des données en Big Data est la protection des infrastructures. De nombreuses technologies développées dans le but d'assurer un calcul de haute performance, tel que l'écosystème Hadoop, ne disposent pas de protections de sécurité adéquates [Pathak, 17]. Par ailleurs, l'IoT fait référence à des réseaux hautement interconnectés

d'appareils hétérogènes où toutes sortes de communications semblent possibles, même celles non-autorisées. Dans un tel contexte, l'exigence de sécurité est critique [Pathak, 17].

La Big Data fournit des solutions de collecte, de stockage et de traitement de données très puissantes améliorant considérablement l'exploitation des données et l'utilisation des ressources informatiques. Cependant, les défis de sécurité en termes de cryptographie, d'analyse des journaux et des événements, de la détection et de la prévention des intrusions et du contrôle d'accès ont pris une nouvelle dimension en Big Data [Sudarsan, 15]. Ainsi, nous dédions une grande partie de ce chapitre à l'étude détaillée de l'un des principaux mécanismes de protection des données, à savoir le contrôle d'accès. Il s'agit d'une mesure importante pour la protection des données, des services et des ressources d'un système d'information afin d'empêcher les utilisateurs illégitimes d'accéder aux objets protégés et les utilisateurs légitimes d'accéder à des objets en dehors de ce qui leur est autorisé. Un modèle de contrôle d'accès comprend essentiellement quatre éléments: le sujet, l'objet, l'action et la politique de contrôle d'accès. Le sujet représente toute entité active manipulant des données (un utilisateur interne ou externe du système, une application, un processus système, …). L'objet représente toute entité passive contenant des données requises dans les traitements du sujet (répertoire, fichier, table d'une base de données, …), des services (service web, page web, …) ou des ressources du système (bande passante, serveur, disque de stockage, CPU, mémoire vive, …). L'action représente les opérations que le sujet peut effectuer sur un objet (lecture, écriture, suppression, écriture sans lecture, exécution, …). La restriction placée sur l'accès d'un sujet à un objet est déterminée par la politique d'accès qui représente l'ensemble des règles d'accès (qui accède à quoi, comment et dans quelles circonstances et conditions). Pour être complet, un système de contrôle d'accès doit comprendre trois composants [Pinno, 17] : l'authentification, l'autorisation et l'audit. L'authentification identifie la bonne identité du sujet. L'autorisation vérifie si le sujet a le droit d'effectuer une opération. Enfin, l'audit permet l'analyse a posteriori des activités réalisées dans le système.

Les modèles de contrôle d'accès existent depuis le début des années 1970. Le contrôle d'accès a été initialement introduit pour résoudre le problème de l'autorisation d'accès aux données partagées sur un mainframe[23]. Le contrôle d'accès discrétionnaire (DAC, pour Discretionary Access Control) et obligatoire (MAC, pour Mandatory Access Control) sont ainsi apparus [Bell, 76]. DAC a l'avantage de la flexibilité, mais n'est pas adapté aux données à grande échelle avec des exigences de sécurité élevées en raison de ses propriétés de gestion décentralisée des ressources et de gestion complexe des autorisations. MAC peut résoudre le problème causé par la décentralisation de la gestion des ressources, mais souffre du problème d'une gestion trop stricte des autorisations. Pour un système avec un grand nombre d'utilisateurs et de nombreux types d'informations et de ressources qui ne sont pas clairement définis, MAC pourrait entraîner une charge de travail excessive, une faible efficacité et un manque de flexibilité [Cai, 18]. Avec l'évolution des technologies de l'information, les modèles de contrôle d'accès DAC et MAC ne peuvent plus répondre aux besoins des applications modernes. En conséquence, des modèles de contrôle d'accès basés sur les rôles (RBAC, pour Role-based Access Control) ont émergé. RBAC peut traiter efficacement le problème de sécurité causé par la flexibilité de DAC et la limitation de MAC mais lui-même présente de nombreuses limites face aux exigences de la Big Data, notamment la prise en charge du contexte d'accès. Aujourd'hui, les modèles de contrôle d'accès basés sur les attributs (ABAC, pour Attribute-based

[23] Un mainframe est un ordinateur central de grande puissance de traitement qui sert d'unité centrale à un réseau de terminaux. Ce système informatique est censé disparaitre avec l'évolution du Cloud Computing et de la Big Data.

Access Control) sont de plus en plus adoptés par les chercheurs notamment pour le stockage de données sur le Cloud.

Dans ce chapitre, nous examinons les modèles et les politiques de contrôle d'accès dans différents scénarios d'applications. Une attention particulière est accordée au contrôle d'accès pour les domaines d'applications de la Big Data tels que les environnements distribués et collaboratifs, le Cloud Computing et les systèmes IoT. Nous étudions également les systèmes de contrôle d'accès implémentés dans la plateforme Hadoop et son écosystème ainsi que certains modèles de contrôle d'accès qui les étendent.

Le reste de ce chapitre est organisé comme suit. La section 2 se concentre sur la présentation des principaux défis de sécurité en Big Data. Nous y discutons les grands défis pour assurer la confidentialité, l'intégrité et la disponibilité de données ainsi que la protection de la vie privée des individus. La section 3 présente les grandes phases composant le cycle de vie Big Data ainsi que les principaux risques de sécurité pour chaque phase. La section 4 est dédiée à la présentation des mécanismes de cryptographie, largement utilisés dans les systèmes de protection de données et les modèles de contrôle d'accès. Nous y présentons également la blockchain qui est une technique émergente très utilisée pour la protection des systèmes IoT. La section 5 est réservée à la présentation des modèles de contrôle d'accès classiques (DAC, MAC, RBAC et ABAC). Ensuite, dans la section 6, nous présentons les langages de contrôles d'accès, notamment XACML et NGAC qui sont les plus adoptés dans les travaux de recherche. Nous dédions ensuite la section 7 à la présentation des modèles et des systèmes de contrôle d'accès en Big Data ; cette section est organisée selon quatre domaines : les environnements distribués et collaboratifs, les environnements de stockage sur le Cloud et le Cloud Computing, les environnements intelligents et les systèmes IoT et, enfin, les plateformes Big Data à travers le cadre Hadoop et son écosystème. Nous discutons enfin ce chapitre dans la section 8 et le concluons dans la section 9.

3.2. Sécurité de données dans le contexte Big Data

Par ordre décroissant d'importance, la sécurité s'est traditionnellement concentrée sur la confidentialité, l'intégrité et la disponibilité de données [Sudarsan, 15]. A cela, nous ajoutons la protection de la vie privée des individus comme étant un objectif majeur auquel tout projet Big Data doit apporter une attention particulière. Dans cette section, nous allons rappeler la définition de chacune de ces notions et présenter les principaux défis et problèmes à relever dans le contexte Big Data.

3.2.1. Confidentialité

La confidentialité est un sujet très discuté en matière de sécurité de données. Il s'agit d'appliquer un ensemble de règles et de restrictions dans le but de limiter l'accès aux données aux entités autorisées. La confidentialité est souvent assurée par des mécanismes de contrôle d'accès qui reposent sur une ou plusieurs techniques telles que [Sudarsan, 15] :

- le stockage de données en texte brut et ensuite les chiffrer lors de leur transition,
- le stockage de données en texte brut en exigeant une authentification pour en avoir accès,
- ou encore, le chiffrement de données avant de les stocker et les déchiffrer lors de l'accès.

Cependant, le chiffrement de données dans un environnement Big Data est limité par la complexité de gestion d'un nombre élevé de clés. En outre, chiffrer des grandes masses de données ne semble pas être une tâche facile à réaliser ; les opérations de chiffrement et de déchiffrement peuvent non seulement ralentir l'ensemble du système mais aussi impacter l'intégrité de données. Dans [Bertino, 15], Bertino adresse de nombreux défis à relever pour assurer la confidentialité de données en Big Data que l'on peut résumer en cinq points :

- **Fusion et intégration des politiques de contrôle d'accès :** les données sont collectées à partir de multiples sources et peuvent être associées à leurs propres politiques de contrôle d'accès. Dans de nombreux cas, ces politiques peuvent être persistantes et doivent être appliquées ce qui peut engendrer des situations conflictuelles avec d'autres politiques d'accès. Il faudra donc intégrer les politiques de sécurité persistantes et résoudre les éventuels conflits en utilisant un système d'intégration automatisé.
- **Conception, évolution et gestion automatique des stratégies de contrôle d'accès** : dans un environnement dynamique où les sources, les utilisateurs, les applications et l'utilisation des données changent continuellement, la capacité de concevoir, d'évoluer et de gérer automatiquement les stratégies d'accès est essentielle pour assurer la disponibilité de données tout en garantissant leur confidentialité.
- **Administration automatique des autorisations** : le contrôle d'accès à granularité fine est une exigence requise, mais son application manuelle sur des grands ensembles de données n'est pas évidente. Des techniques par lesquelles les autorisations peuvent être accordées automatiquement, éventuellement en fonction de l'identité numérique, du profil et du contexte de l'utilisateur ainsi que du contenu et des métadonnées des données doivent être mises en œuvre.
- **Application des autorisations à des données multimédia hétérogènes** : le contrôle d'accès basé sur le contenu consiste à accorder ou refuser les autorisations en fonction du contenu des données. Lorsqu'il s'agit des données volumineuses multimédias, non-structurées, la compréhension du contenu des données constitue un vrai défi.
- **Application du contrôle d'accès sur les plateformes Big Data** : les plateformes Big Data ne permettent pas une application simple des politiques de contrôle d'accès. Par exemple, dans Hadoop, les utilisateurs peuvent soumettre des travaux MapReduce dans différents langages de programmation ce qui crée des défis importants pour appliquer efficacement un contrôle d'accès à granularité fine.

Par ailleurs, la tendance aujourd'hui est d'héberger les données sur des Clouds externes pour profiter de la souplesse que le Cloud Computing offre comme la mise à l'échelle, le paiement à l'utilisation (pay-as-you-use), la robustesse et la fiabilité des infrastructures. Ce nouveau paradigme d'hébergement de données pose des défis sérieux à la confidentialité notamment sur les endroits physiques et les régions géographiques où les données sont stockées ainsi que les entités qui peuvent accéder aux données sans autorisations explicites de leurs propriétaires.

Enfin, une nouvelle tendance qui prend de plus en plus de la place dans les marchés d'exploitation des informations est la propriété mutuelle des données. Pour des raisons de supériorité et de compétitivité dans les entreprises, différentes organisations coopéreront sous la forme de partage de données et se donnent le droit d'accès aux données des uns par les autres ce qui accentue le risque de fuite de données. Une approche pour répondre à ces préoccupations est basée sur des systèmes collaboratifs distribués par lesquels les organisations conservent leurs propres ensembles de données et coopèrent pour apprendre les

résultats globaux de l'exploration de données sans révéler les données dans leurs propres ensembles de données individuels [Pathak, 17].

3.2.2. Intégrité

L'intégrité signifie que les données n'ont pas été modifiées et sont identiques à l'heure à laquelle elles ont été créées ou éditées par des entités autorisées. L'intégrité implique la cohérence de données entre les différentes copies. Une solution clé pour assurer la haute disponibilité des données est de créer plusieurs copies ce qui peut présenter un vrai défi pour assurer l'intégrité. D'autres problèmes peuvent survenir et impacter l'intégrité comme les erreurs matérielles et logicielles, les erreurs humaines ou les intrusions [Sudarsan, 15]. Pour maintenir l'intégrité de données, il faut gérer plusieurs processus :

- **Fiabilité des sources de données** : sans informations de provenance, il peut être impossible pour un utilisateur de savoir d'où les données proviennent et quelles transformations ont pu se produire, ce qui peut nuire à la valeur de ces données [Azmi, 15]. La provenance de données fait référence aux informations sur l'origine et le processus de création de données. Ces informations sont utiles pour l'audit, l'évaluation de la qualité et de la fiabilité de données, la modélisation de l'authenticité et la mise en œuvre du contrôle d'accès pour les données dérivées [Glavic, 14].
- **Fiabilité de données** : une des principales utilités de la Big Data est l'aide à la prise de décision. Pour que les utilisateurs produisent une analyse précise et prennent des mesures, des prédictions et des décisions efficaces, les données doivent être fiables. Garantir la fiabilité de données est un problème difficile, car il faut s'assurer non seulement que les données soient correctes, mais aussi que les données soient protégées contre les parties malveillantes visant à tromper les utilisateurs. Le problème est encore plus complexe par le fait qu'une solution spécifique à adopter pour assurer la fiabilité de données dépend souvent de la sémantique du domaine d'application considéré [Bertino, 15].
- **Prévention de la perte de données** : la perte, le vol ou la fuite de données est un problème impactant l'intégrité de données. L'utilisation d'une approche de "Cloud Privé" peut conduire à une meilleure gestion de données mais ne peut empêcher la perte de données due aux risques de feu, inondation, tremblement de terre, coupure électrique, piratage et mort subite des disques durs (en raison de la chaleur, de la durée de vie ou d'une panne du courant électrique). En revanche, le maintien de l'accessibilité, du contrôle et de la gestion des données requiert une stratégie à long terme et une solution complète, y compris un plan de sauvegardes régulières de données et de reprise après sinistre pour assurer la restauration rapide de données [Chang, 15]. La réplication géographique de données protège également contre la perte de données [Gray, 05].
- **Déduplication de données** : outre la perte de l'espace de stockage et le ralentissement de la bande passante du trafic réseau, la redondance de données a un grand impact sur l'intégrité de données. Lorsqu'il y a plusieurs copies d'une même donnée, les mécanismes de synchronisation des modifications apportées à la donnée deviennent plus complexes. Zhou et al. [Zhou, 13] identifient trois sources de redondance en Big Data, à savoir le déploiement de plusieurs nœuds de stockage de données, l'expansion des ensembles de données et l'adoption des mécanismes de réplication de données. La déduplication de données est une technique de stockage qui permet de factoriser les séquences de données identiques et d'éliminer ainsi les redondances afin d'économiser l'espace de stockage et de libérer la bande passante du trafic réseau. Elle peut être également utilisée lors de l'archivage de données pour gagner de l'espace de stockage des archives. La technique de

déduplication consiste à découper chaque fichier en plusieurs blocs et à associer un identifiant unique à chaque bloc. L'objectif est de ne stocker qu'une seule copie d'un bloc ; toute nouvelle occurrence déjà présente n'est pas à nouveau sauvegardée mais remplacée par une référence vers l'identifiant correspondant.

3.2.3. Disponibilité

La disponibilité peut être définie comme étant la propriété d'un système à être accessible et utilisable par les entités autorisées [Zissis, 10]. Elle fait référence à la capacité du système à continuer à fonctionner même en cas de violation ou lorsque certaines fonctionnalités sont en erreurs. La disponibilité concerne les données, les logiciels et le matériel. Pour assurer la disponibilité de données dans un contexte Big Data, un système doit répondre à un ensemble de critères :

- **Tolérance aux intrusions et aux erreurs** : un système tolérant aux intrusions et aux erreurs est un système qui reste opérationnel même en cas d'intrusion ou d'erreur matérielle, logicielle ou humaine. Une intrusion peut concerner non seulement les accès non-autorisés mais aussi les abus de privilèges des utilisateurs autorisés. Généralement, dans un environnement Big Data, les données sont stockées dans un système distribué et réparties sur plusieurs nœuds. Comme le risque d'intrusion, de défaillance d'un nœud ou d'un logiciel est important, la tolérance aux intrusions et aux erreurs est primordiale.
- **Mise à l'échelle (Scalabilité)** : en Big Data, le volume de données augmente très rapidement et dépasse les capacités des ressources informatiques et les vitesses des processeurs [Khan, 14]. Pour assurer le stockage et le traitement de ces volumes croissants de données, le système doit être extensible que ce soit à l'échelle matérielle ou logicielle. Au niveau de l'infrastructure, le système doit pouvoir accepter l'ajout et la suppression des nœuds de stockage et de traitement avec des modifications limitées, voire automatiques, de la configuration. Au niveau logiciel, les algorithmes de traitement de données, les mécanismes d'équilibrage de la charge (Load Balancing) sur les différents nœuds ainsi que les processus de calculs parallèles doivent s'adapter automatiquement aux évolutions de l'infrastructure.
- **Récupération de données** : une des principales caractéristiques de la disponibilité est la capacité de récupérer les données en cas de perte. La sauvegarde et la réplication de données sont des mécanismes qui permettent de récupérer les données en cas de défaillance matérielle ou logicielle ou suite à une erreur humaine. Cependant, ces mécanismes posent de nombreux problèmes concernant la duplication de données. En effet, la réplication de données peut provoquer des incohérences entre différentes copies d'une même donnée en cas de problème matériel ou logiciel. Le processus de réplication ne doit donc pas être au détriment de la cohérence. Par ailleurs, la sauvegarde de données, si elle n'est pas bien conçue, pourrait ralentir la bande passante du trafic réseau et épuiser l'espace de stockage ce qui peut engendrer une latence dans tout le système.
- **Surveillance et Audit** : la surveillance et l'audit sont deux processus de sécurité qui aident à détecter les intrusions et les comportements anormaux. Ils consistent notamment à surveiller et à analyser les activités des utilisateurs et du système, à analyser les configurations et les vulnérabilités du système, à analyser les activités anormales, à vérifier le respect de la charte utilisateurs, etc. Ces deux processus se basent principalement sur le balayage de toutes les données et de tout le trafic ce qui est particulièrement complexe dans un environnement Big Data.

3.2.4. Protection de la vie privée

La protection de la vie privée est une notion complexe et difficile à définir en raison de la subjectivité culturelle des individus liée à ce sujet. Pour certains, elle repose sur des droits fondamentaux tels que la liberté et la propriété [Moore, 08]. Depuis la réservation de l'article 12 de la déclaration universelle des droits de l'homme de 1948 au respect de la vie privée [ONU, 48], plusieurs lois et règlements ont été mis en place comme le RGPD (Règlement Général de Protection des Données) [CNIL, 18-a] ou encore la loi LPRPDE (Loi sur la Protection des Renseignements Personnels et les Documents Electroniques) [LPRPDE, 18]. Selon la LPRPDE, une information personnelle concerne toute information factuelle ou subjective, consignée ou non, concernant une personne identifiable.

Le développement de la Big Data est toujours confronté à de nombreux défis dans lesquels les risques de sécurité et de protection de la vie privée sont reconnus comme l'un des problèmes les plus complexes. Par exemple, l'historique des achats, les dossiers médicaux, les photos et les vidéos personnelles, les revenus financiers, les orientations religieuses et politiques, la géolocalisation, les déplacements, etc., sont tous sous le contrôle d'autres organismes. De toute évidence, la fuite de ces informations sensibles peut avoir des conséquences graves. De nombreux domaines fondés sur les technologies Big Data tels que l'e-commerce, les réseaux sociaux et les environnements intelligents ont accentué cette problématique. Les propriétaires de données ne savent généralement pas exactement où elles sont stockées, par qui, qui peut les exploiter, de quelle façon et pour quelles fins. Aucune garantie sur la protection de données personnelles n'est garantie. Par exemple, en avril 2018, Facebook a accusé la société "Cambridge Analytica" d'avoir accédé, de façon indue, aux données de 87 millions d'utilisateurs [meta, 18]. En février 2021, plus de 3.2 milliards de combinaisons d'e-mails et de mots de passe pour des comptes (Netflix, Gmail, Hotmail, LinkedIn,…) ont été publiées en ligne [cybernews, 21] ; surnommée "Compilation De Nombreuses Violations" ou COMB (Compilation Of Many Breaches), cette opération concerne des informations d'identification qui avaient été volées dans le cadre de violations précédentes et de fuites d'entreprises comme Netflix et LinkedIn.

La protection de la vie privée consiste à contrôler le partage des informations personnelles identifiables. Elle implique différents processus de sécurité tels que :

- **Identification des données personnelles** : l'ICO (Information Commissioner's Office) définit les données personnelles comme étant des données qui se rapportent à un individu vivant identifiable. Le terme "identifiable" signifie que l'individu peut être identifié à partir de ces données, seules ou en combinaison avec d'autres informations [ICO, 14]. La combinaison d'un ensemble d'informations permet donc d'identifier les individus d'où la nécessité non seulement de déterminer les données personnelles mais aussi les liens avec toutes autres informations susceptibles de déduire l'identité d'un individu. Aujourd'hui, avec l'énorme volume de données massives telles que les médias sociaux qui contiennent énormément d'informations personnelles hautement interconnectées, chaque information sur tout le monde peut être exploitée ; quand toutes les informations sur une personne sont extraites et rassemblées, tout le cadre privé concernant cet individu disparaît [Che, 13].
- **Dés-identification et prévention de la ré-identification** : la dés-identification est une technique d'extraction de données préservant la vie privée. Elle consiste à stocker et à partager les données sans révéler l'identité des personnes impliquées en remplaçant les quasi-identifiants par des valeurs moins spécifiques mais sémantiquement cohérentes. Généralement, les techniques de dés-

identification interviennent à trois niveaux de protection de la vie privée [Cavoukian, 14]. Premièrement, elles protègent les enregistrements d'un individu d'être identifiés de manière unique dans l'ensemble de données. Deuxièmement, elles empêchent les enregistrements d'un individu d'être liés à d'autres ensembles de données. Et, troisièmement, elles rendent difficile la déduction d'informations sensibles sur un individu à partir de l'ensemble de données dés-identifié. Cependant, même si des informations personnelles identifiables sont supprimées, lorsque certaines données sont combinées avec d'autres données, un individu peut être identifié. Il s'agit essentiellement du problème d'inférence et d'agrégation que les chercheurs en sécurité de données explorent depuis quatre décennies [Toshniwal, 15]. Ce problème est exacerbé avec le contexte Big Data car il existe désormais différentes sources de données liées aux individus [Pathak, 17]. En Big Data, un attaquant peut obtenir des informations externes ce qui peut augmenter le risque de ré-identification [Lu, 14]. La ré-identification est le processus d'attribution des données personnelles à l'identité de leur propriétaire. Les attaques de ré-identification ont trois sous-catégories : les attaques de corrélation, les attaques d'identification arbitraires et les attaques d'identification ciblées. Ce type d'attaques consiste à balayer des grands ensembles de données pour construire les corrélations conduisant à une empreinte unique d'un individu [Matturdi, 14].

- **Identification des propriétaires de données** : pour tout système, l'accès aux données est contrôlé de façon manuelle, semi-automatique ou automatique par l'administrateur ou le propriétaire de données. En Big Data, il n'est pas facile de connaitre le (ou les) propriétaire(s) d'une donnée. Par exemple, une photo d'une personne peut être la propriété de la personne prise en photo, de la personne ayant pris la photo et/ou de la personne l'ayant créée dans le système. En outre, une donnée peut être la propriété de plusieurs personnes qui peuvent avoir des objectifs différents, voire contradictoires. Pour résoudre tous ces conflits sur la propriété de données, des solutions technologiques, organisationnelles et juridiques doivent être étudiées [Bertino, 15]. Les travaux de Damen et al. [Damen, 14] abordent cette problématique et considèrent que les données dans les environnements collaboratifs, comme les réseaux sociaux, sont créées et maintenues par des utilisateurs différents, selon des capacités différentes, et proposent un modèle permettant à chacun des propriétaires d'exprimer ses règles d'accès selon son point de vue et sa capacité.
- **Préserver les résultats d'analyse de données** : les problèmes de protection des données personnelles continuent d'entraver les individus et deviennent encore plus complexes avec le développement des technologies Big Data Mining (exploration des données volumineuses). Ces technologies utilisent et analysent des informations personnelles pour produire des résultats pertinents, tels que des services personnalisés comme par exemple les publicités ciblées et individualisées [Che, 13]. En l'absence d'une protection effective de ces résultats, la vie privée des individus serait impactée.
- **Conformité aux contraintes réglementaires et légales** : la Big Data peut être appliquée dans différents secteurs comme la santé, la cyber-sécurité, la sûreté, la protection des territoires, le e-commerce, etc. Pour ces domaines d'application, l'accès aux données personnelles est souvent réglementé par des lois qui autorisent, sous certaines conditions, d'accéder et d'analyser les informations personnelles des individus. A titre d'exemple, nous pouvons citer la loi HIPAA (Health Insurance Portability and Accountability Act) de 1996 [HIPAA, 96] appliquée dans le domaine de la santé aux États-Unis où l'utilisation et la divulgation des informations personnelles est autorisée à condition de satisfaire aux exigences de la loi sur la transférabilité et la responsabilité en matière d'assurance maladie. Un autre exemple concerne un projet de loi, en

cours d'étude, lancé par la Commission Européenne dans le cadre de la lutte contre le terrorisme permettant l'autorisation de la collecte des données personnelles par la police dans toute l'Europe [Reuters, 17]. L'objectif de cette nouvelle loi est de faciliter les enquêtes policières en permettant aux forces de l'ordre de récupérer les données des réseaux d'entreprises et des réseaux sociaux.

3.3. Risques de sécurité au long du cycle de vie Big Data

Comme illustré dans la Figure 3.1, le cycle de vie d'un projet Big Data est généralement composé de quatre phases principales [Alladi, 18]:

- la collecte de données auprès des sources internes et externes d'une organisation.
- le stockage de données dans des environnements distribués et évolutifs.
- l'analyse, l'exploration et le traitement distribués de données.
- la création de connaissances pour générer de la valeur.

Certains travaux, comme [Bao, 18], exigent une phase de destruction de données à l'issue de l'extraction de connaissances.

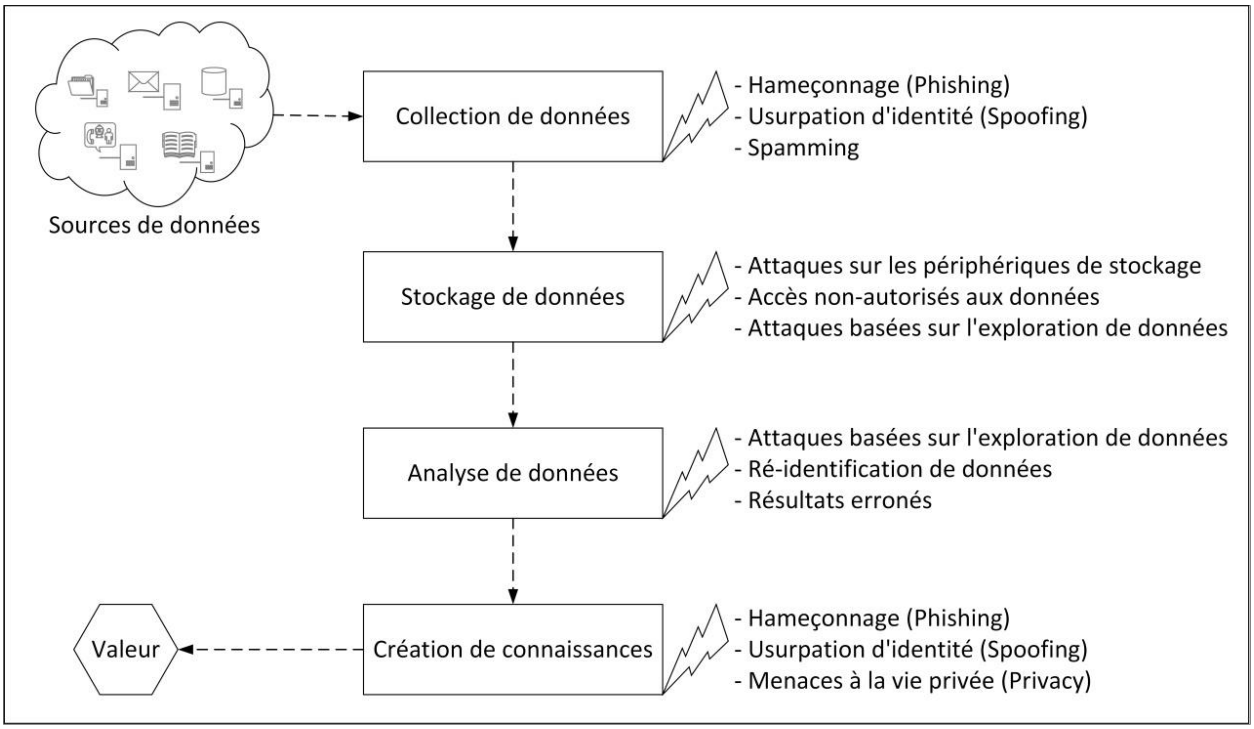

Figure 3.1. Risques de sécurité pendant le cycle de vie Big Data (adapté de [Alladi, 18])

Dans le reste de cette section, nous allons détailler les risques de sécurité lors de chaque phase du cycle de vie Big Data.

3.3.1. Collection de données

Cette phase consiste à récupérer les informations depuis une diversité de sources de données. Pour assurer la sécurité de données, il est essentiel de collecter les données à partir de sources de données fiables et de s'assurer qu'elles sont bien protégées. Il faudra également sécuriser les modes de transfert de données sur le réseau. Toutefois, les pirates utilisent de nombreuses techniques pour collecter les données confidentielles. Parmi ces techniques on peut citer :

- **Hameçonnage** (Phishing) : cette technique consiste à envoyer un courriel par un cybercriminel déguisé en courriel provenant de source légitime et digne de confiance. Le contenu du courriel vise à inciter le destinataire à révéler des informations sensibles ou confidentielles.
- **Usurpation d'identité** (Spoofing) : cette technique consiste à usurper l'identité d'un individu ou d'une organisation dans le but de recueillir des données confidentielles.
- **Spamming** : un spam est un message non-sollicité qui peut arriver sous forme d'email, de message instantané ou encore de fichier média (vidéo, audio ou image). Ces messages, assez faciles à repérer, peuvent endommager le support de stockage de données.

De bonnes pratiques, des formations, de la sensibilisation aux risques de fuite de données et des contrôles doivent être mis en œuvre par les organisations pour lutter contre ce type d'attaques.

3.3.2. Stockage de données

Une fois collectées, les données doivent être stockées et protégées afin d'assurer un environnement sécurisé pour leur analyse. La plupart des systèmes de stockage Big Data existants aujourd'hui répondent aux exigences de stockage et d'évolutivité (stockage et traitement distribués, robustesse, fiabilité, etc.), mais ne garantissent pas la concurrence et les besoins de sécurité [Bao, 18]. Par ailleurs, lors du passage sur le Cloud, il n'y a aucune garantie quant à l'endroit physique où les données seront stockées et qui accédera à ces données [Dissanayake, 21]. Les administrateurs des serveurs de stockage, en particulier sur le Cloud, dotés d'une haute autorité peuvent s'entendre pour exploiter des données précieuses. Contrôler certains des gestionnaires ayant une haute autorité pour les données et les mots de passe est également un défi [Bao, 18]. Ainsi, les principales menaces et attaques de sécurité pouvant atteindre les données stockées sont :

- **Attaquer les disques de stockage de données** : voler ou effectuer des copies des disques.
- **Accès non-autorisés** : accès aux données de façon illégale ou abus de privilèges, notamment les hébergeurs de données.
- **Attaques basées sur l'exploration de données** (Data Mining) : cibler et explorer les données pour en extraire des connaissances.

3.3.3. Analyse de données

L'analyse de données est effectuée pour extraire des informations importantes en appliquant des techniques d'exploration de données telles que la classification, le regroupement ou encore les règles d'association. Pour l'analyse de données, le chiffrement est un moyen efficace pour assurer la sécurité de données. Toutefois, dans un environnement Big Data, l'énorme quantité de données pourrait augmenter la

charge de chiffrement. En effet, pour les applications Big Data en temps réel, telles que les réseaux sociaux, les environnements intelligents, etc., l'exigence de la rapidité pour l'analyse des flux de données est un problème rendant plus complexes les méthodes de chiffrement et de protection de données. Les principales menaces et attaques de sécurité pouvant survenir lors de l'analyse de données sont :

- **Attaques basées sur l'exploration de données** : cibler et explorer les données pour en extraire des connaissances et des informations masquées ou cachées.
- **Ré-identification de données** : cette technique consiste à analyser des données disponibles publiquement ou piratées pour rechercher des corrélations menant à une empreinte unique d'un individu. Plus précisément, en reliant différents types d'ensembles de données, l'unicité de chaque entrée est augmentée, jusqu'à ce qu'un lien vers l'identité d'un individu puisse être établi.
- **Résultats erronés** : les résultats dépendent des requêtes effectuées par l'analyste de données. Par exemple, si un ensemble de données a été formé par corrélation d'adresses électroniques contenues dans tous les jeux de données corrélés, on peut supposer que deux entrées contenant la même adresse électronique avant la corrélation conduiront à une entrée de base de données unique reflétant toutes les informations d'identité. Cependant, dans certains cas, cette hypothèse peut ne pas tenir. Par exemple, certaines adresses e-mail peuvent être partagées par deux personnes ou plus, provoquant un faux couplage de données si elles sont utilisées pour la corrélation avec des individus [Jensen, 13].

3.3.4. Création de connaissances

Les informations obtenues suite à l'analyse de données sont très sensibles. Elles peuvent être appliquées par les décideurs pour améliorer la compétitivité de leurs organisations. La protection de ces données est obligatoire. Les menaces de sécurité pendant cette phase concernent principalement :

- **Hameçonnage et Usurpation d'identité** : les pirates peuvent cibler les décisions prises à l'issue de l'analyse de données en utilisant des techniques d'attaques comme l'hameçonnage et l'usurpation d'identité.
- **Menaces de la vie privée** : l'analyse de données peut porter sur des informations personnelles des individus accessibles de façon directe (accès autorisé aux données) ou indirecte (par exemple, en corrélant plusieurs informations). Les résultats d'analyse de ces données personnelles concernent donc la vie privée des individus et, s'ils ne sont pas bien protégés, la vie privée de ces individus sera menacée. En outre, afin de préserver la vie privée des individus, certaines données sensibles pourraient être détruites du Cloud. D'ailleurs, certaines réglementations, comme le RGPD [CNIL, 18-a], exigent la destruction des données après utilisation. Cependant, à notre connaissance, très peu d'études traitent les défis de sécurité au stade de la destruction de données.

3.4. Cryptographie : fonctions et applications

La CNIL (Commission Nationale de l'Informatique et des Libertés) définit la cryptologie comme étant la science du secret [CNIL, 16]. Elle réunit la cryptographie, qui fait référence à un ensemble de mécanismes pour réaliser des écritures secrètes, et la cryptanalyse qui correspond à l'étude des attaques contre les mécanismes de cryptographie. Dans cette section, nous nous intéressions à la cryptographie qui, aujourd'hui, assure non-seulement la confidentialité des données secrètes mais s'est élargie à prouver

mathématiquement d'autres notions telles que l'authenticité d'un message (prouver qui est l'auteur d'un message) ou encore son intégrité (prouver s'il a été modifié ou non). Pour assurer ces usages, la cryptographie regroupe les trois principales fonctions suivantes:

- le hachage
- le chiffrement et le déchiffrement
- la signature numérique

Dans cette section nous allons présenter brièvement ces trois fonctions ainsi que deux concepts fondamentaux, à savoir le certificat électronique et la blockchain. Il est nécessaire de connaitre ces notions de base pour la bonne compréhension du reste de ce chapitre.

3.4.1. Hachage

Le hachage permet de garantir qu'un message est intègre et de détecter s'il a été involontairement modifié. Comme le montre la Figure 3.2, une "fonction de hachage" permet d'associer à un message M (un simple texte, un fichier texte ou multimédia, un répertoire, etc.), quelle que soit sa taille, une empreinte unique $H(M)$, qui est une suite de caractères hexadécimaux de taille fixe. La taille de l'empreinte dépend de la fonction de hachage utilisée. Connaissant l'algorithme de hachage :

- toute personne peut calculer l'empreinte $H(M)$ d'un message M. Cette valeur ne change pas tant que la fonction de hachage n'a pas changé.
- personne ne peut retrouver le message M à partir de son empreinte $H(M)$; il s'agit d'une fonction à sens unique (irréversible).

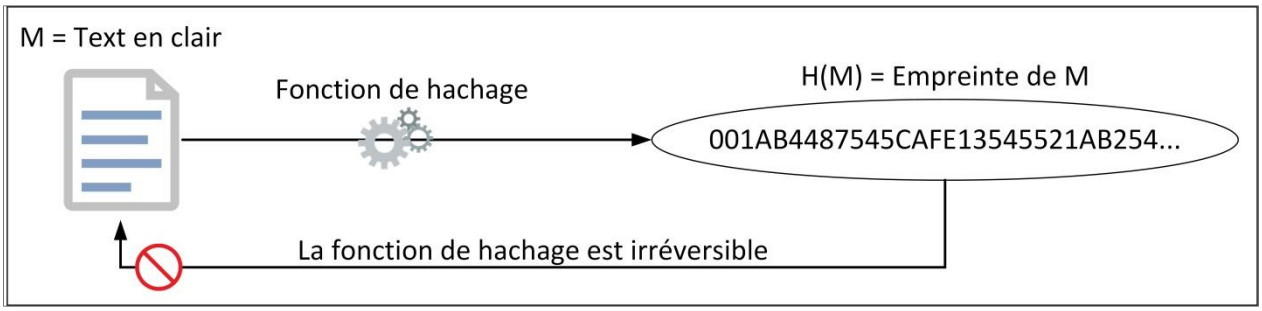

Figure 3.2. Mécanisme de hachage

L'empreinte $H(M)$ doit être une caractéristique du message M et il doit y avoir une très faible probabilité de collision, c'est-à-dire que deux messages différents aient la même empreinte. Différents algorithmes existent pour calculer l'empreinte d'un message :

- **MD5** (Message Digest 5) : inventée par Ronald Rivest [Rivest, 92], cette fonction de hachage produit une empreinte de 128 bits (16 octets ou 32 caractères en notation hexadécimale). Wang et al. ont démontré qu'il était possible d'obtenir la même empreinte (des collisions) sur des fichiers différents [Wang, 04], ce qui fait donc de MD5 un algorithme faillible et obsolète (même s'il est encore un standard très utilisé dans les entreprises pour le stockage des mots de passe).
- **SHA-1** (Secure Hash Algorithm 1) : conçue par la NSA (National Security Agency), cette fonction de hachage produit une empreinte de 160 bits (20 octets ou 40 caractères en notation

hexadécimale). En 2005, des cryptanalystes ont découvert des attaques sur SHA-1 déduisant ainsi que l'algorithme pourrait ne plus être suffisamment sûr [Schneier, 05].

- **SHA-2** (Secure Hash Algorithm 2) : est une famille de fonctions de hachage qui ont été conçues par la NSA, sur le modèle de la fonction SHA-1. Cette famille comporte de nombreuses fonctions dont les plus utilisées aujourd'hui sont SHA-256 et SHA-512 qui ont des algorithmes similaires mais opèrent sur des tailles de mot différentes (256 bits pour SHA-256 et 512 bits pour SHA-512). On y trouve également d'autres fonctions comme SHA-384 qui est considérée plus sécurisée que SHA-512 et apporte certains changements à l'initialisation et à la sortie (tronquée à 384 bits).
- **SHA-3** (Secure Hash Algorithm 3) : en octobre 2012, le NIST (National Institute of Standards and Technology) a annoncé la sélection de l'algorithme Keccak de Bertoni et al. [Bertoni, 08] comme lauréat du concours d'algorithmes de hachage cryptographique SHA-3. Cette fonction n'est pas destinée à remplacer SHA-2 mais à fournir une autre solution à la suite des possibilités d'attaque contre les standards MD5 et SHA-1.

Outre l'intégrité numérique, le hachage a de nombreuses applications telles que la signature numérique, le certificat électronique, le stockage des mots de passe, la gestion des blocs dans une blockchain, etc.

3.4.2. Chiffrement et Déchiffrement

Le chiffrement et le déchiffrement (ou, cryptage et décryptage) sont un procédé de cryptographie qui permet de rendre la compréhension d'un message impossible à toute personne qui n'a pas la clé de déchiffrement. Le chiffrement consiste, grâce à une clé et un algorithme de chiffrement, à transformer un texte en clair en un texte chiffré, appelé cryptogramme, qui n'est pas compréhensible. Le déchiffrement est l'opération inverse ; grâce à une clé et un algorithme de déchiffrement, il permet de transformer un texte chiffré en texte en clair. Trois types de chiffrement existent : le chiffrement symétrique, le chiffrement asymétrique et le chiffrement hybride.

Le chiffrement symétrique, comme illustré dans la Figure 3.3, permet de chiffrer et de déchiffrer un message moyennant une clé secrète dite "clé privée". C'est la même clé qui est utilisée pour chiffrer et déchiffrer le message.

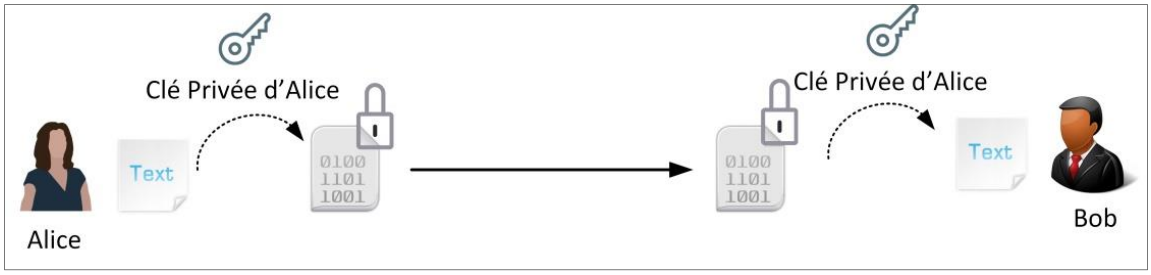

Figure 3.3. Mécanisme du chiffrement symétrique

Le chiffrement symétrique est particulièrement rapide, puisqu'on utilise la même clé, mais nécessite que l'émetteur et le destinataire se mettent d'accord sur une clé secrète commune ou se la transmettent par un autre canal [CNIL, 16]. Celui-ci doit être choisi avec précaution pour éviter que la clé soit récupérée par des pirates.

Le chiffrement asymétrique, quant à lui, est basé sur deux clés : une pour le chiffrement et une autre pour le déchiffrement. Si la clé de chiffrement est publique, celle de déchiffrement doit être secrète et vice versa. Comme le chiffrement symétrique utilise la même clé, il est plus rapide que le chiffrement asymétrique qui nécessite des calculs plus complexes pour chiffrer et déchiffrer des données. Dans le cas général, comme le montre la Figure 3.4, le chiffrement asymétrique consiste à ce que l'émetteur utilise la clé publique du destinataire pour chiffrer le message et que le destinataire utilise sa clé privée pour le déchiffrer.

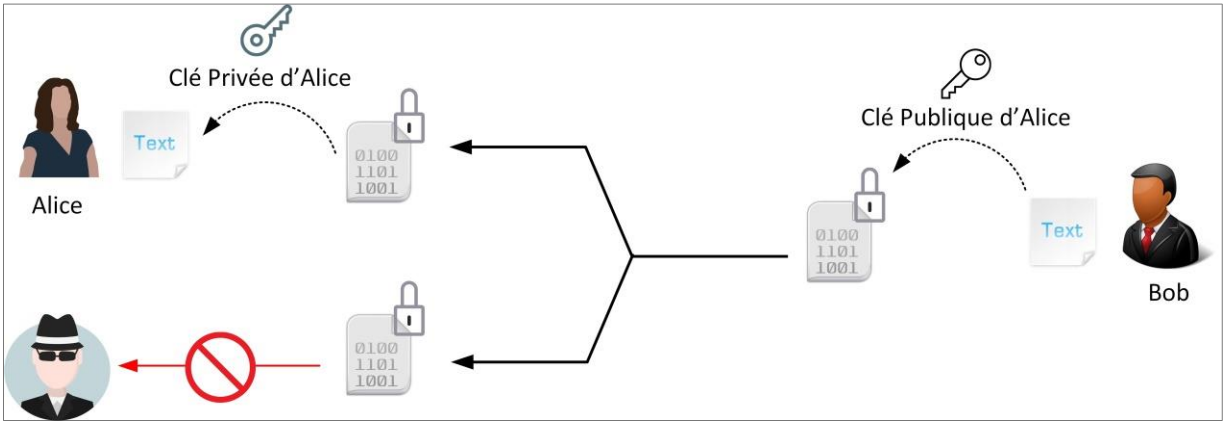

Figure 3.4. Chiffrement asymétrique (chiffrement par clé publique et déchiffrement par clé privée)

Dans certaines situations, comme le montre la Figure 3.5, il est possible que l'émetteur chiffre un message avec sa clé privée et que le destinataire utilise la clé publique de l'émetteur pour déchiffrer le message. Ce processus est utilisé par exemple pour prouver l'authenticité de l'émetteur d'une signature électronique.

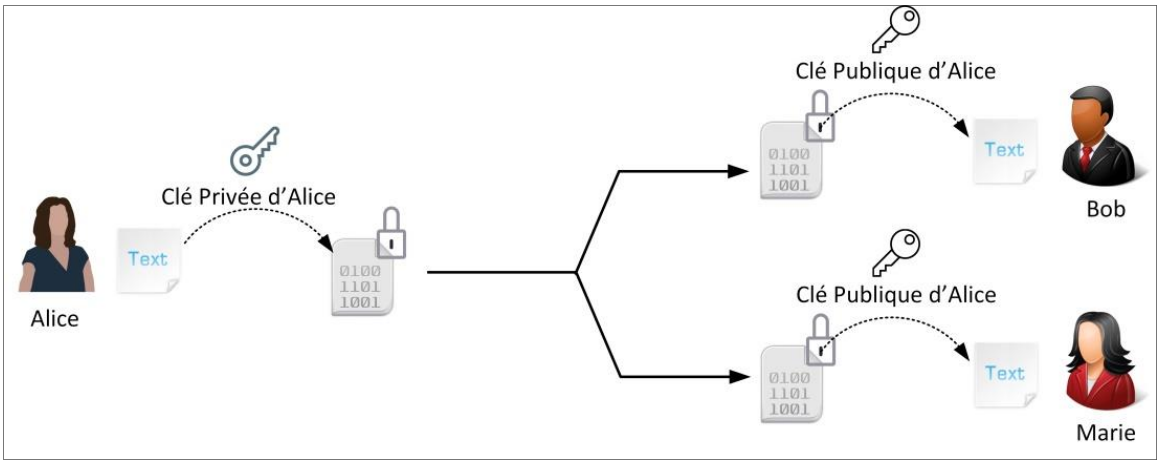

Figure 3.5. Chiffrement asymétrique (chiffrement par clé privée et déchiffrement par clé publique)

L'avantage du chiffrement asymétrique c'est qu'il n'y a pas de clé secrète à se partager entre l'émetteur et le destinataire : connaissant la clé publique du destinataire, l'émetteur chiffre le message, et ensuite, le destinataire, étant le seul à connaitre sa clé privée, l'utilise pour déchiffrer les messages qu'il reçoit. Le risque dans ce mode de fonctionnement c'est que l'émetteur doit s'assurer que la clé publique

est bien celle du destinataire et qu'elle n'a pas été remplacée par celle d'un pirate. Grâce au certificat électronique, présenté un peu plus loin, ce problème d'authenticité de la clé publique est résolu.

Le chiffrement hybride est une technique qui combine le chiffrement symétrique et asymétrique dans le but de bénéficier de leurs avantages. Ce type de chiffrement consiste à se partager la clé secrète chiffrée par un chiffrement asymétrique, et ensuite, une fois connue par l'émetteur et le destinataire, l'utiliser pour chiffrer et déchiffrer les messages confidentiels par un chiffrement symétrique qui présente l'avantage d'être rapide.

Concernant les algorithmes de chiffrement, l'AES (Advanced Encryption Standard) [FIPS PUBS, 01], aussi connu par l'algorithme Rijndael, est aujourd'hui le standard le plus utilisé dans le chiffrement symétrique. L'AES est destiné à remplacer le DES (Data Encryption Standard) [FIPS PUBS, 99] qui est devenu trop faible au regard des attaques d'aujourd'hui. Pour le chiffrement asymétrique, RSA [RSA, 78], nommé par les initiales de ses trois inventeurs Rivest, Shamir et Adleman, est l'algorithme le plus utilisé aujourd'hui pour échanger des données confidentielles sur Internet.

3.4.3. Signature Numérique

Par analogie avec la signature manuscrite d'un document papier, la signature numérique permet d'authentifier l'auteur d'un message (ou de tout document électronique) et de garantir son intégrité (en s'assurant qu'il n'a pas été modifié). Une signature numérique doit répondre à trois caractéristiques : l'authenticité (elle ne doit pas pouvoir être imitée), la non-répudiation (le signataire ne doit pas pouvoir se rétracter ou prétendre qu'il ne l'a pas signée) et l'intégrité (en comparant la signature et le message, on doit pouvoir être sûr que le message n'a pas été modifié lors de l'envoi). La signature numérique est basée sur le chiffrement asymétrique et les fonctions de hachage selon le protocole illustré dans la Figure 3.6.

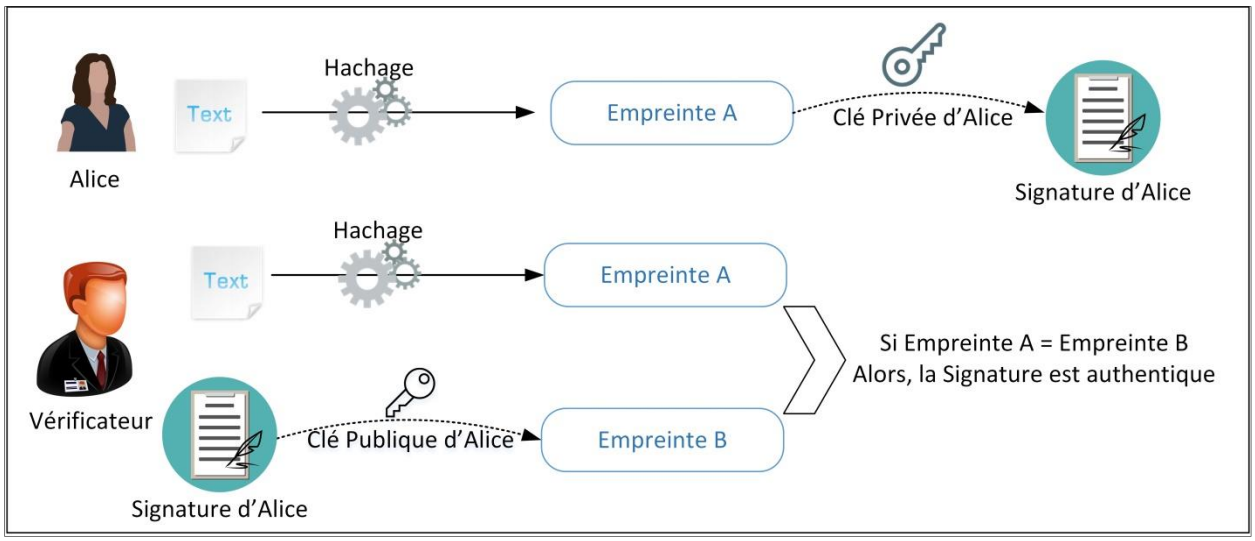

Figure 3.6. Mécanisme de la signature numérique

Le protocole de la signature numérique consiste à :

- Pour signer un message, le signataire doit, tout d'abord, générer l'empreinte du message au moyen d'une fonction de hachage et, ensuite, chiffrer cette empreinte avec sa clé privée. La signature ainsi obtenue, est transmise avec le message au destinataire.
- Pour vérifier la validité du message, le vérificateur doit tout d'abord générer l'empreinte du message en utilisant la même fonction de hachage appliquée par le signataire et, ensuite, déchiffrer la signature en utilisant la clé publique du signataire. Il suffit enfin de comparer les deux empreintes pour juger de la validité de la signature numérique.

3.4.4. Certificat Electronique

Le chiffrement asymétrique est basé sur le partage d'une clé publique entre différents utilisateurs. Le partage de cette clé publique se fait souvent au travers d'un annuaire ou d'un serveur web. Le problème c'est que dans ce mode de partage, rien ne garantit que la clé publique est bien celle de l'utilisateur qui l'a partagée. Un pirate qui réussit à remplacer une clé publique par la sienne pourra déchiffrer tous les messages ayant été chiffrés par cette nouvelle clé corrompue. Les certificats électroniques permettent de résoudre ce problème. Un certificat électronique permet d'associer une clé publique à une entité (une personne, une organisation, une machine, ...) afin d'en assurer la validité. Ainsi, lorsqu'on récupère une clé publique, grâce au certificat électronique qui lui est associé, on saura si elle est valide ou pas.

Le certificat électronique peut être vu comme la pièce d'identité de l'entité désirant partager une clé publique. Il est généralement délivré par un organisme appelé "autorité de certification" ou CA (pour Certification Authority). Une autorité de certification est chargée de délivrer les certificats et de leur assigner une date de validité. Elle peut éventuellement révoquer à tout moment un certificat électronique. Selon la norme X509[24], le certificat électronique doit contenir un ensemble d'informations telles que le nom de l'autorité de certification l'ayant délivré, sa date de validité, la clé publique associée, certaines informations sur l'entité ayant partagé la clé publique (nom, prénom, adresse électronique, etc.), …, et la signature de l'autorité de certification. Cette signature électronique est une empreinte générée à partir des informations contenues dans le certificat (nom, adresse, clé publique, …). La signature est chiffrée par la clé privée de l'autorité de certification ayant délivré le certificat.

Selon le niveau de signature, on distingue deux types de certificats :

- **Les certificats auto-signés** : ce sont des certificats signés par un serveur local et destinés à un usage interne. Ce type de certificats permet de garantir la confidentialité des échanges au sein d'une organisation.
- **Les certificats signés par un organisme de certification** : pour assurer la confidentialité des échanges avec des utilisateurs externes (par exemple les sites de e-commerces, le paiement en ligne, les échanges confidentiels avec des organismes publics ou privés, etc.), il est nécessaire que les certificats soient délivrés par des autorités de certifications.

Les certificats électroniques peuvent être utilisés dans différents contextes ; voici quelques exemples :

- Le certificat installé sur un serveur web permet d'assurer le lien entre un service et son propriétaire. Par exemple, dans le cas d'un site web, le certificat permet de garantir que le nom de

[24] https://www.itu.int/rec/T-REC-X.509/fr

domaine dans l'url d'une page web appartient bien au propriétaire du site web. Il permet également de sécuriser les échanges avec les utilisateurs grâce aux protocoles HTTPS (Hypertext Transfer Protocol Secure), SSL (Secure Socket Layer) et TLS (Transport Layer Security).

- Un certificat installé sur le poste de travail d'un utilisateur ou embarqué dans une carte à puce permet d'identifier l'utilisateur et de lui associer ainsi des droits.
- Un certificat installé sur les équipements d'un réseau virtuel privé VPN (Virtual Private Network) permet de chiffrer les flux de communication de bout en bout entre deux points (par exemple, l'accès au réseau interne d'une organisation depuis l'extérieur).

3.4.5. Blockchain

Avec son émergence dans les crypto-monnaies, la blockchain a été conçue pour servir de base de données distribuée sécurisée pour le stockage des transactions d'actifs numériques[25]. Il s'agit d'un registre dans lequel les transactions sont enregistrées dans des blocs chainés dans un ordre chronologique d'où sa nomination de "blockchain". Ce registre est public, décentralisé, tolérant aux fautes, immuable et ne dépend d'aucune autorité centrale. La blockchain est désignée comme étant une technologie qui a surmonté avec succès deux problèmes complexes :

- Sans l'intervention d'une partie centralisée, atteindre un consensus dans un groupe de participants anonymes dont certains peuvent se comporter avec une intention malveillante.
- Eviter les attaques de sécurité, comme le problème de la double-dépense[26], et assurer l'intégrité de toutes les informations sur la blockchain.

La blockchain a eu son progrès dans le Bitcoin [Nakamoto, 08] pour enregistrer de façon sécurisée les transactions financières. Aujourd'hui, elle est utilisée pour stocker tout type d'informations. A titre d'exemples, Ouaddah et al. [Ouaddah, 17] l'utilisent pour stocker des contrats intelligents appliqués pour échanger des exécutions de politiques de contrôle d'accès contre des jetons d'accès ; Pinno et al. [Pinno, 17] l'utilisent pour stocker les règles de contrôle d'accès, les relations, les contextes et les informations de responsabilité ; Azaria et al. [Azaria, 16] l'utilisent pour contrôler les autorisations d'accès aux données des dossiers médicaux des patients à travers des contrats intelligents entre les patients, les prestataires et les tiers pour accorder des autorisations d'accès, etc. Certains modèles de contrôle d'accès, que nous étudions un peu plus loin dans ce chapitre, sont basés sur la blockchain ; c'est pour cette raison que nous dédions cette section à la présentation des notions fondamentales de la blockchain.

La blockchain est une technologie qui permet de stocker de façon sécurisée des données numériques dans une architecture distribuée. Elle est basée principalement sur le hachage. Comme son nom l'indique, une blockchain est une chaine de blocs ordonnés, chacun contient des données numériques en plus du hash du bloc précédent. Le tout premier bloc s'appelle le bloc de genèse (genesis block, en anglais) et sert de base sur laquelle des blocs supplémentaires sont ajoutés pour former la chaîne de blocs. Dans une blockchain, un nouveau bloc ne peut être créé que si le hash du bloc précédent ait été généré (sachant que le hash d'un bloc ne peut être calculé qu'à la fin d'écriture de toutes les données qu'il doit contenir). Comme illustré dans la Figure 3.7, le hash du bloc 64 n'a été calculé qu'après avoir terminé toutes les

[25] Un actif numérique est un bien immatériel, constitué de données numériques, qui possède une valeur financière déterminée par l'offre et la demande, comme tout autre actif financier.
[26] https://en.bitcoin.it/wiki/Irreversible_Transactions

écritures de ce bloc et le bloc 65 n'a été généré qu'après avoir calculé le hash du bloc 64. C'est donc la fonction de hachage qui permet d'ordonner les blocs dans une blockchain.

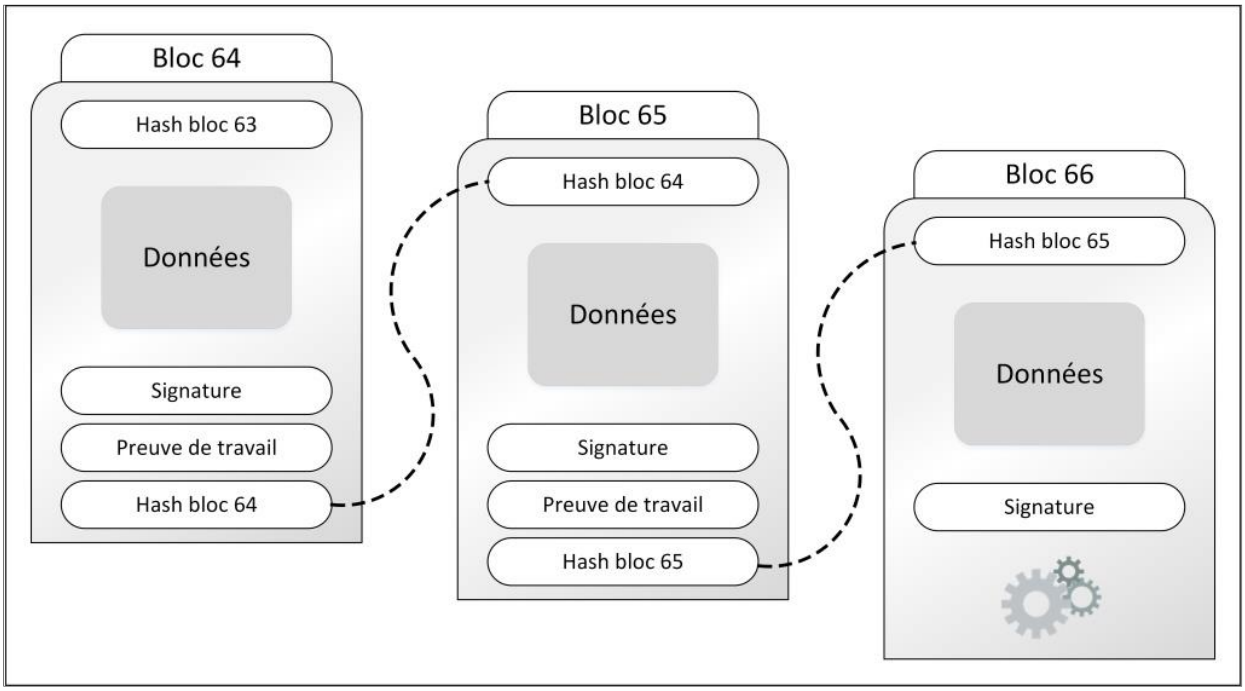

Figure 3.7. Principe de la Blockchain

Un bloc peut contenir toute sorte de données (des transactions dans le cadre de la crypto-monnaie, des données que l'on doit protéger ou archiver, des données que l'on souhaite partager dans un réseau distribué, etc.). L'écriture dans un bloc suit généralement les étapes suivantes :

- Tout d'abord, il doit contenir le hash du bloc précédent.
- Ensuite, on y inscrit les données que l'on souhaite écrire. On peut éventuellement y ajouter les signatures des auteurs.
- Et enfin, on calcule le hash du bloc pour pouvoir créer le bloc suivant.

Pour mieux expliquer la robustesse de la blockchain, prenons l'exemple de la crypto-monnaie Bitcoin [Nakamoto, 08]. Il s'agit d'un réseau créé en 2009 par Satoshi Nakamoto (c'est le pseudonyme adopté par la ou les personnes ayant développé le Bitcoin). Concrètement, cette blockchain regroupe un ensemble de personnes sur un réseau. Chaque personne est représentée par un nœud, possède un portefeuille de bitcoins, peut télécharger une copie à jour de tous les blocs de la blockchain et peut valider les transactions. Toutes les transactions réalisées entre les différents nœuds du réseau sont inscrites dans des blocs. Les blocs sont générés (ou minés) par des "mineurs" selon un consensus bien défini. Ce processus s'appelle le "minage" et consiste à :

1. récupérer toutes les transactions non encore inscrites sur la blockchain : à chaque fois qu'un nœud du réseau effectue une transaction (transfert de sommes vers d'autres nœuds), celle-ci est propagée dans tout le réseau de la blockchain. La transaction n'est effectivement valable qu'une fois validée par un mineur.
2. vérifier les transactions : cela consiste à vérifier la signature électronique associée à chaque transaction et que l'émetteur est bien en possession des fonds qu'il désire transmettre.

3. inscrire les transactions dans un bloc : la taille d'un bloc ne dépasse pas 1 Mo (généralement, un bloc est généré toutes les 10 minutes en moyenne).
4. calculer "une preuve de travail" et l'ajouter dans le bloc : c'est l'étape la plus complexe dans le processus de minage ; elle consiste à résoudre un problème mathématique très difficile pour trouver une valeur, appelée "nonce" (Number Only Used Once), répondant à certaines exigences.
5. partager le bloc, comprenant la preuve de travail représentée par le nonce, avec les autres mineurs pour validation et inscription sur la blockchain contre une rémunération prédéterminée en Bitcoin.

Comme il y a plusieurs mineurs sur la blockchain, ils se mettent tous en compétition pour calculer la preuve de travail en résolvant des énigmes (puzzles) mathématiques difficiles. Le mineur qui trouve la preuve de travail en premier en informe les autres. Si plusieurs mineurs trouvent la réponse en même temps, ce qui est théoriquement possible, c'est celui qui a le plus de blocs attachés postérieurement à son bloc qui gagne ; c'est le consensus de la blockchain Bitcoin !

Concrètement, le calcul de la preuve de travail consiste à calculer l'empreinte du bloc en utilisant les informations qui y sont inscrites (transactions, signatures, …) et le nonce. Une règle de la blockchain est qu'on ne peut ajouter un bloc à la blockchain que si son empreinte commence par un certain nombre prédéterminé de zéros. Comme il n'est pas évident que le résultat du hachage commence par le nombre de zéros nécessaires et comme on ne peut pas modifier les informations inscrites sur le bloc, les mineurs ajustent le nonce pour trouver une empreinte commençant par le nombre de zéros requis. Cet ajustement consiste à modifier la valeur du nonce, recalculer l'empreinte et répéter ce processus indéfiniment jusqu'à trouver une bonne valeur du nonce générant l'empreinte contenant le nombre minimal de zéros nécessaires à l'inscription du bloc dans la blockchain.

Le calcul de la preuve de travail nécessite un très grand nombre d'itérations : plus le nombre d'itérations est grand, plus le temps pour trouver la réponse est lent d'où l'intérêt d'avoir des calculateurs très performants et très énergivores pour pouvoir trouver rapidement la réponse. L'idée de la "preuve de travail" a été développée pour la première fois en 1993 par Cynthia Dwork et Moni Naor [Dwork, 93] et a été formalisée en 1999 par Markus Jakobsson et Ari Juels [Jakobsson, 99]. Le but derrière une preuve de travail est de démontrer que beaucoup de temps et beaucoup d'énergie ont été dépensés dissuadant ainsi les personnes malveillantes de tenter des attaques de sécurité. Pour calculer la preuve de travail, le Bitcoin adopte l'algorithme Hashcash, conçu par Adam Back en 1997 [Back, 97], [Back, 02]. Cet algorithme, tel qu'il est utilisé dans le Bitcoin, consiste à trouver une collision partielle de la fonction de hachage dSHA-256[27]. Cela revient à trouver deux messages ayant une empreinte commençant par les mêmes caractères. Comme la fonction de hachage est à sens unique, il faudra tester plusieurs millions (ou milliards) de combinaisons pour trouver les messages. Dans Bitcoin, la preuve de travail consiste à trouver un message dont l'empreinte calculée par la fonction dSHA-256 commence par un certain nombre de zéros prédéterminés ; ce message doit également comporter des indications sur le contexte dans lequel la preuve de travail a été calculée (identifiant du mineur, timestamp de calcul, etc.) pour démontrer qu'elle n'a pas été utilisée autre part.

Pour la gestion des transactions, les blockchains peuvent être globalement classées en deux catégories :

[27] dSHA-256, ou double SHA-256, ressemble à la fonction SHA-256 mais appliquée 2 fois de suite.

- **les blockchains basées sur la sortie de transaction non dépensée - UTXO** (Unspent Transaction Output) : dans ce type de blockchains, chaque transaction correspond à un transfert de pièces entre plusieurs nœuds (relations plusieurs-à-plusieurs). Un transfert consomme (i.e. dépense des pièces à partir de) certaines entrées et crée (i.e. dirige les pièces vers) de nouvelles sorties. Il s'agit donc d'un graphe dirigé dans le temps reliant les opérations de transferts entre les différents nœuds représentés par leurs adresses publiques dans le réseau de la blockchain.
- **les blockchains basées sur le compte** : dans ce type de blockchains, un nœud, représenté par son adresse publique, peut dépenser une fraction de ses pièces et conserver le solde restant. Dans ces blockchains, une transaction a exactement une entrée et une adresse de sortie (relations un-à-un contrairement à UTXO). Bien que la création d'adresse soit gratuite, une seule adresse est généralement utilisée pour recevoir et envoyer des pièces plusieurs fois.

3.5. Modèles de contrôle d'accès classiques

L'objectif du contrôle d'accès est de déterminer si un sujet peut appliquer une action à un objet. Le sujet (une personne, une application, un processus, etc.) représente toute entité active pouvant manipuler des données. L'objet (un répertoire, un fichier, une table d'une base de données, un disque de stockage, un service web, etc.) représente toute entité passive contenant les données requises dans les traitements du sujet. L'action (lire, écrire, supprimer, transférer, etc.) représente les opérations que le sujet peut effectuer sur un objet. Un modèle de contrôle d'accès comprend donc quatre composants: le sujet, l'objet, l'action et la politique de sécurité. Cette dernière représente l'ensemble des règles d'accès permettant aux sujets d'appliquer des actions aux objets. On distingue deux grandes familles de politiques de sécurité : les politiques discrétionnaires DAC (Discretionary Access Control) et les politiques obligatoires MAC (Mandatory Access Control). Avec l'évolution des technologies de l'information, ces politiques ont atteint leurs limites et de nouvelles variantes ont été proposées parmi lesquelles on trouve les politiques basées sur les rôles RBAC (Role-Based Access Control) et les politiques basées sur les attributs ABAC (Attribute-Based Access Control). Toutefois, la majorité des modèles de contrôle d'accès proposés dans la littérature ne répondent pas aux contraintes des environnements Big Data. Nombreux d'entre eux tentent souvent d'étendre les modèles RBAC et/ou ABAC pour remplir certaines exigences.

Dans la suite de cette section, nous allons présenter brièvement les modèles de contrôle d'accès DAC, MAC, RBAC et ABAC. L'objectif étant de présenter les prérequis à la bonne compréhension des modèles de contrôle d'accès en Big Data que nous présenterons dans la section suivante.

3.5.1. Modèles de contrôle d'accès discrétionnaires (DAC)

Les modèles de contrôle d'accès discrétionnaires donnent à l'utilisateur la responsabilité de définir le niveau de protection de ses objets, en attribuant ou en retirant les droits d'accès aux autres sujets du système. Ces modèles sont fondés sur la notion de propriété : le propriétaire d'un objet est le sujet qui l'a créé. A titre d'exemple, le système de gestion des fichiers UNIX repose sur un modèle DAC pour la gestion des droits d'accès : le propriétaire d'un fichier peut librement définir des droits (Lecture, Ecriture, Exécution) pour lui-même, pour son groupe ou pour d'autres utilisateurs. Parmi les modèles qui garantissent le contrôle d'accès discrétionnaire, nous citons :

- **Modèle Lampson** [Lampson, 74] : ce modèle permet la spécification des droits d'accès à l'aide d'une matrice. Il est représenté par le triplet (S, O, M) où S représente le Sujet, O représente l'Objet et M représente la Matrice de contrôle d'accès associant à chaque couple (S, O) un ensemble de droits d'accès. La Table 3.1 représente un exemple d'une matrice d'accès.

Table 3.1. Exemple d'une matrice de contrôle d'accès selon le modèle Lampson

	Objet O_1	Objet O_2	…	Objet O_m
Sujet S_1	Lire, Ecrire, Exécution	Ecrire	…	Lire
Sujet S_2	Ecrire	Lire	…	Ecrire
…	…	…	…	…
Sujet S_n	Lire, Ecrire	Exécution	…	Lire

La matrice de contrôle d'accès selon le modèle Lampson est composée d'un ensemble fini de sujets, d'objets cibles et d'actions autorisées. Le modèle Lampson ne permet donc pas de spécifier de manière explicite les interdictions d'accès ; la matrice conduit à l'établissement d'une liste exhaustive d'autorisations d'accès ACL (Access Control List) ce qui implique que tout accès non explicitement permis est interdit.

- **Modèle HRU** [Harrison, 76] : portant le nom de ses créateurs (Harrison, Ruzzo et Ullman), ce modèle apporte une amélioration au modèle Lampson et fournit des commandes permettant non seulement l'attribution des droits d'accès mais aussi la création et la suppression des sujets et des objets. Une règle du modèle HRU est définie par un quadruplet {S, O, R, M} où :
  - S représente l'ensemble des sujets,
  - O représente l'ensemble des objets,
  - R représente l'ensemble des modes d'accès (lecture, écriture, exécution, …, possession).
  - M représente la matrice d'accès.

  Le mode d'accès "Possession" signifie que le sujet étant le "Propriétaire" de l'objet. Ce mode d'accès n'est ni attribuable, ni révocable. Tous les autres modes d'accès peuvent être attribués ou révoqués par les propriétaires.

Le contrôle discrétionnaire est désavantagé par sa vulnérabilité au Cheval de Troie, ce qui présente un risque important de fuite d'informations (le propriétaire d'un objet peut attribuer à un sujet malicieux l'accès à des informations confidentielles). En plus de leur vulnérabilité, les modèles discrétionnaires présentent des difficultés d'expression comme l'impossibilité d'exprimer des règles en dépendance avec le contexte du travail. Ils présentent également des difficultés de structuration qui nécessite une mise à jour de la politique de contrôle d'accès lors de l'ajout de nouveaux sujets ou la création de nouveaux objets. Malgré toutes ces limites, les modèles discrétionnaires sont utilisés dans de nombreuses applications modernes comme Facebook. Celui-ci donne le droit aux utilisateurs de définir les autorisations relatives aux informations qu'ils publient.

3.5.2. Modèles de contrôle d'accès obligatoires (MAC)

Pour limiter le pouvoir de gestion des accès, les modèles de contrôle d'accès obligatoires consistent à affecter aux sujets et aux objets un ensemble d'attributs de sécurité non modifiables par les utilisateurs. Ce modèle est utilisé lorsque la politique de sécurité du système d'information impose que le propriétaire d'un objet n'a aucun rôle de prise de décision en termes de protection de son objet ; toutes les décisions de

protection sont imposées par le système de sécurité lui-même. Parmi les modèles qui assurent le contrôle d'accès obligatoire, nous citons :

- **Sécurité multi-niveaux** : le principe de la sécurité multi-niveaux consiste à attribuer à chaque entité du système des niveaux d'habilitation et de classification. Le contrôle d'accès d'un sujet S à un objet O est fait non seulement en fonction de l'identité du sujet S mais aussi en fonction de son habilitation par rapport au niveau de classification de l'objet O. L'habilitation et la classification reflètent les niveaux de sécurité respectifs aux sujets et aux objets. A titres d'exemples, nous pouvons avoir les niveaux de sécurité suivants :
  - "*Non Classé*" : ce niveau s'applique uniquement aux informations qui peuvent circuler librement à l'extérieur d'une organisation et ne justifient pas de protection particulière.
  - "*Usage Interne*" : ce niveau regroupe les informations qui peuvent circuler librement à l'intérieur d'une organisation.
  - "*Diffusion Restreinte*" : ce niveau de sécurité s'applique aux informations qui ne doivent être communiquées qu'aux personnes directement concernées et identifiées précisément.
  - "*Secret*" : ce niveau est réservé aux informations dont la divulgation à des personnes non-autorisées pourrait porter atteinte aux intérêts stratégiques de l'organisation.
- **LBAC** : le contrôle d'accès basé sur les treillis, en anglais Lattice Based Access Control, a été proposé par Sandhu [Sandhu, 93]. Un treillis, en mathématiques, est un ensemble partiellement ordonné dans lequel chaque paire d'éléments admet une borne supérieure et une borne inférieure. Le modèle LBAC est fondé sur un treillis avec le concept d'ordre partiel. Une relation d'ordre partiel est binaire, réflexive, antisymétrique et transitive sur un ensemble et permet de comparer les éléments de cet ensemble entre eux de manière cohérente. Dans le cadre du contrôle d'accès, on définit une relation d'ordre partiel par trois éléments:
  - une classification linéaire des niveaux de sécurité H (Habilitation),
  - une catégorisation C, et
  - des étiquettes de sécurité (h, c) où $h \in H$ et $c \subseteq C$.

  Dans LBAC, une étiquette de sécurité définie par la paire (h_i, c_i) peut être liée à une autre étiquette par une relation d'ordre partiel (dom) : *(h1, c1) dom (h2, c2) si et seulement si* $h1 \leq h2$ *et* $c1 \subseteq c2$
- **Modèle de Bell et La Padula** [Bell, 76] : ce modèle consiste à assurer la confidentialité des données en :
  - interdisant toute fuite d'information d'un objet possédant une certaine classification vers un objet possédant un niveau de classification inférieur.
  - interdisant à tout sujet possédant une certaine habilitation d'obtenir des informations d'un objet d'un niveau de classification supérieur à cette habilitation. Un sujet S avec une certaine habilitation h(S) ne peut obtenir des informations d'un objet O dont la classification c(O) est supérieure à h(S).

  Formellement : *(S, O, Lire)* $\rightarrow$ $h(S) \geq c(O)$.
- **Modèle de Biba** [Biba, 77] : ce modèle consiste à assurer l'intégrité des données en :
  - interdisant toute propagation d'information d'un objet situé à un certain niveau d'intégrité vers un objet situé à un niveau d'intégrité supérieur.
  - interdisant à tout sujet situé à un certain niveau d'intégrité de modifier un objet possédant un niveau d'intégrité supérieur. Un sujet S avec une certaine habilitation h(S) ne peut écrire dans un objet O dont la classification c(O) est supérieure à h(S).

  Formellement : *(S, O, Ecrire)* $\rightarrow$ $c(O) \leq h(S)$.

Le contrôle obligatoire résout le problème de fuite d'information du contrôle discrétionnaire et limite la diffusion de l'information. Il sépare la gestion des droits d'accès de l'accès lui-même et limite les risques de sécurité dus à des attaques de type Cheval de Troie. Toutefois, ce type de contrôle ne permet pas de gérer les exceptions entre les différents niveaux de sécurité. Par exemple, un utilisateur de niveau de sécurité "*Diffusion Restreinte*" ne peut accéder, pour des raisons exceptionnelles, à la donnée de niveau de sécurité "*Secret*". Il s'agit donc d'un modèle sévère qui impose des contraintes fortes où les informations circulent de façon unidirectionnelle dans un treillis. Ce type de contrôle d'accès est utilisé par les systèmes Linux tel que SELinux (Security-Enhanced Linux), AppArmor (Application Armor) et Smack (Simplified Mandatory Access Control Kernel).

3.5.3. Modèles de contrôle d'accès basés sur les rôles (RBAC)

Dans le contrôle d'accès à base de rôles, chaque décision d'accès à un système d'information est basée sur le(s) rôle(s) assigné(s) à chaque utilisateur dans ce système. Un rôle découle généralement de la structure d'une organisation en considérant que les utilisateurs exerçant des fonctions similaires peuvent être regroupés sous le même rôle. Un rôle, déterminé par une autorité centrale, associe à un sujet des autorisations d'accès sur un ensemble d'objets. Dans RBAC, la modification des règles de contrôle d'accès ne porte pas sur les utilisateurs mais plutôt sur les rôles. Ainsi, lorsqu'un utilisateur rejoint/quitte une organisation, il suffit de lui assigner/dés-assigner un rôle. Par cette caractéristique, RBAC est considéré comme un système "parfait" pour les entreprises dont la fréquence de renouvellement du personnel est élevée. Ce type de contrôle est également référencé sous le nom de contrôle d'accès non-discriminant.

Différentes variantes du modèle RBAC ont été proposées dans la littérature [Sandhu, 96], [Gavrila, 98], [Ahn, 00]. Le modèle NIST pour RBAC [Ferraiolo, 01] a été adopté en tant que norme nationale américaine 359-2004 par l'ANSI/INCITS[28] (American National Standards Institute / International Committee for Information Technology Standards) le 11 février 2004. Il a été révisé en tant qu'INCITS 359-2012 en 2012. NIST a défini différents niveaux (ou composants) pour le modèle RBAC :

- **RBAC-0** : ce niveau représente le modèle RBAC de base, ou le noyau RBAC, défini via les utilisateurs, les rôles et les autorisations.
- **RBAC-1** : ce niveau inclut RBAC-0 et incorpore des hiérarchies en tant que relation d'ordre partiel entre les rôles où les rôles "supérieurs" héritent de toutes les autorisations des rôles "inférieurs".
- **RBAC-2** : ce niveau inclut RBAC-0 et ajoute des contraintes comme la séparation des fonctions SoD (Separation of Duties) qui peuvent être statiques SSD (Static Separation of Duties) ou dynamiques DSD (Dynamic Separation of Duties). D'autres contraintes existent comme les prérequis ou les contraintes sur les cardinalités.
- **RBAC-3** : RBAC-1 et RBAC-2 sont indépendants l'un de l'autre ; un système peut implémenter l'un sans l'autre. Le niveau RBAC-3 combine RBAC-1 et RBAC-2 pour fournir à la fois des hiérarchies de rôles et des contraintes.

La Figure 3.8 illustre une vue d'ensemble du modèle RBAC.

[28] https://csrc.nist.gov/Projects/Role-Based-Access-Control

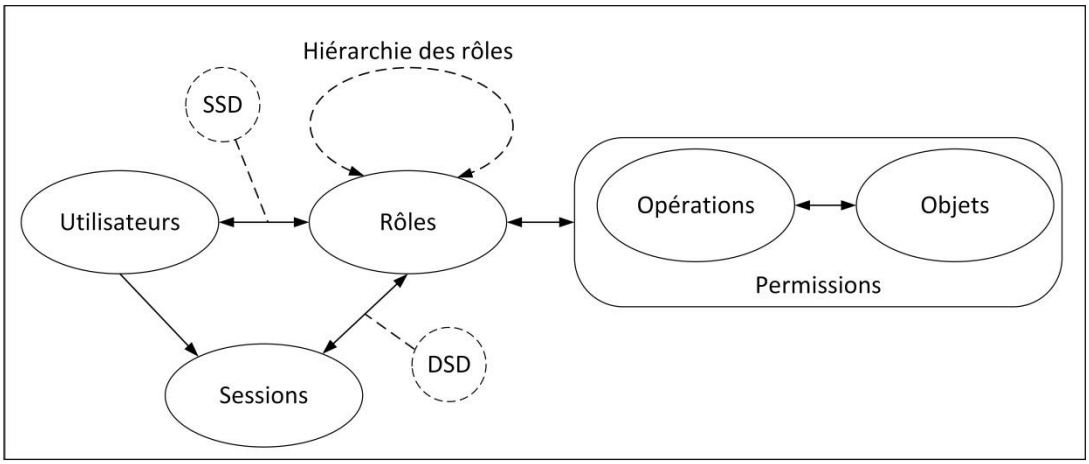

Figure 3.8. Vue d'ensemble du modèle RBAC

Dans la suite de cette section, nous allons détailler chacun des composants du modèle RBAC.

3.5.3.1. Noyau RBAC

Le concept de base du modèle RBAC consiste à ce que des rôles soient assignés à des utilisateurs, les permissions soient assignées à des rôles et les utilisateurs acquièrent des permissions en étant membres dans des rôles. Ce concept englobe les aspects essentiels suivants :

- Un utilisateur peut se voir assigner un ou plusieurs rôles ; et un rôle peut être attribué à un ou plusieurs utilisateurs.
- Une permission peut être assignée à un ou plusieurs rôles ; et un rôle peut être assigné à une ou plusieurs permissions.
- L'activation et la désactivation des rôles se fait dans des sessions.
- Un utilisateur peut exercer simultanément les permissions de multiples rôles.

3.5.3.2. Hiérarchie RBAC

Ce composant ajoute l'exigence hiérarchique entre les rôles. La hiérarchie est un ordre partiel définissant la relation de supériorité entre les rôles par laquelle :

- les rôles supérieurs obtiennent les permissions des rôles inférieurs, et
- les rôles inférieurs obtiennent une adhésion systématique des utilisateurs supérieurs.

Ce critère reconnaît deux types d'hiérarchies de rôles :

- **Hiérarchie générale** : elle permet de supporter le concept de "hiérarchie multiple" qui fournit la capacité d'héritage des permissions de deux rôles ou plus.
- **Hiérarchie restreinte** : elle permet aux systèmes d'imposer des restrictions sur la hiérarchie de rôles. Plus communément, les hiérarchies sont limitées par de simples structures telles que les arbres et les arbres inversés.

3.5.3.3. Contraintes RBAC

Les contraintes constituent un aspect important du modèle RBAC. Elles peuvent être utilisées pour analyser le contrôle d'accès. Les exemples communs des contraintes englobent la séparation statique des

fonctions, la séparation dynamique des fonctions et à cela s'ajoutent les contraintes de prérequis et les contraintes de cardinalité qui ont été introduites dans [Ray, 04].

3.5.3.3.1. Séparation Statique des Fonctions

La contrainte de séparation statique des fonctions (SSD, pour Static Separation of Duties) vise à empêcher le conflit d'intérêts qui apparaît lorsque des rôles en conflit sont associés à un même utilisateur. Cela permet d'empêcher l'affectation de rôles conflictuels à un même utilisateur. Le conflit d'intérêts dans un rôle peut surgir comme le résultat d'un utilisateur obtenant une autorisation pour des permissions liées aux rôles en conflit. Un moyen de prévenir cette forme de conflit d'intérêt est la SSD qui peut être utilisée pour la définition des entités mutuellement exclusives. Pour cela, les contraintes SSD sont spécifiées pour n'importe quelle paire de rôles qui sont en conflit. Les contraintes SSD sont violées lorsqu'on affecte à un utilisateur un ensemble de rôles incluant une paire en SSD, ou lorsqu'un rôle hérite de deux rôles en SSD. La séparation statique des fonctions place les contraintes lors de l'affectation des rôles aux utilisateurs. L'adhésion à un rôle peut empêcher l'utilisateur d'être membre d'un ou plusieurs rôles, en fonction des contraintes SSD appliquées.

3.5.3.3.2. Séparation Dynamique des Fonctions

La séparation dynamique des fonctions (DSD, pour Dynamic Separation of Duties), comme les contraintes SSD, limite les permissions qui sont disponibles pour un utilisateur. Toutefois, les contraintes DSD diffèrent des SSD par le contexte dans lequel ces restrictions sont appliquées. Les contraintes DSD limitent la disponibilité des permissions en imposant des restrictions sur les rôles qui peuvent être activés à l'intérieur d'une session ou entre deux sessions d'un utilisateur. Les propriétés DSD fournissent un support étendu (par rapport au temps) au principe du moindre privilège dans lequel chaque utilisateur possède, à des moments différents, des permissions spécifiques en fonction des tâches et des opérations à exécuter. Cela garantit que les permissions ne se prolongent pas au-delà du temps requis pour l'exécution de leurs fonctions. Les contraintes DSD permettent à un utilisateur d'être autorisé pour des rôles qui n'entraînent pas de conflit d'intérêts, mais qui produisent des violations de la politique quand ils sont activés en même temps.

3.5.3.3.3. Contraintes de prérequis

Le concept de prérequis sur les rôles est basé sur les notions de compétence et de pertinence. Les contraintes de prérequis exigent qu'un utilisateur ne peut être affecté à un rôle que s'il est déjà affecté aux prérequis du rôle [Ray, 04]. Le concept de prérequis peut également être appliqué à d'autres éléments comme la permission dans RBAC. Le concept de prérequis sur les permissions exige qu'une permission ne peut être attribuée à un rôle que si le rôle possède déjà les prérequis de la permission.

3.5.3.3.4. Contrainte de Cardinalité

Les contraintes de cardinalité peuvent être utilisées pour limiter, par exemple, le nombre d'utilisateurs pouvant être affectés à un rôle, le nombre de rôles qu'un utilisateur peut jouer, le nombre de rôles auxquels une permission peut être attribuée ou le nombre de sessions qu'un utilisateur est autorisé à activer simultanément [Ray, 04].

3.5.3.4. Application du modèle RBAC dans le contexte Big Data

Bien qu'il soit efficace et populaire pour les systèmes traditionnels, le modèle RBAC présente plusieurs limites quand il s'agit d'une application Big Data. Tout d'abord, l'implémentation de RBAC consiste à déterminer l'ensemble des rôles à considérer dans la politique de sécurité ce qui peut être qualifié de statique. La spécification en amont de l'ensemble des rôles ne semble pas répondre aux besoins de sécurité d'un environnement qui change en continu. En outre, l'activation d'un rôle dans une session ne prend pas en considération le contexte global de l'activité d'un utilisateur ou l'état du système. L'importance du contexte lors de l'activation, la désactivation et la gestion des permissions a été identifiée lorsque les systèmes de sécurité ont été classés comme systèmes de sécurité passifs ou actifs [Tolone, 05]. Un système de sécurité passif, comme dans RBAC, est un système qui sert principalement à maintenir les attributions des permissions aux rôles. Un système de sécurité actif, en revanche, prend en compte l'impact du contexte lorsqu'il émerge avec la progression de l'activité. Enfin, RBAC traditionnel n'a pas la possibilité de spécifier un contrôle précis sur, d'une part, des utilisateurs précis ayant certains rôles et, d'autre part, sur des objets ou des éléments d'objets (par exemple, certains enregistrements dans une table d'une base de données). Dans un environnement Big Data, il est insuffisant d'avoir des autorisations de rôle sur des objets ou même en fonction de groupe d'objets. Il arrive souvent qu'un utilisateur ayant un rôle quelconque ait besoin d'une autorisation spécifique sur un élément bien précis. De nombreux modèles étendant RBAC ont été développés dans la littérature pour répondre à ces problématiques. On y trouve notamment le modèle OT-RBAC [Gupta, 17] qui reprend le système de contrôle d'accès implémenté dans le noyau Hadoop et son écosystème, en particulier Apache Ranger[29], et exploite les mérites de RBAC en y ajoutant des attributs d'objet, appelés Tags. Dans le domaine des réseaux sociaux, on trouve le modèle CollAC [Damen, 14] qui repose sur les notions de base de RBAC et permet à plusieurs utilisateurs de définir différentes politiques de sécurité sur un même objet, du point de vue de chacun, tout en distinguant à quel niveau de capacité chaque utilisateur peut appliquer sa politique. Dans la littérature, on trouve de nombreux autres modèles étendant RBAC pour répondre à des besoins spécifiques ; nous reviendrons sur certains de ces modèles un peu plus loin dans ce chapitre.

3.5.4. Modèles de contrôle d'accès basés sur les attributs (ABAC)

Traditionnellement, le contrôle d'accès était basé sur l'identité d'un utilisateur demandant l'exécution d'une opération sur un objet, soit directement (DAC et MAC), soit via des types d'attributs prédéfinis tels que des rôles ou des groupes affectés à cet utilisateur (RBAC). Les praticiens ont noté que cette approche de contrôle d'accès est souvent lourde à gérer étant donné la nécessité d'associer des capacités directement aux utilisateurs ou à leurs rôles ou groupes [Hu, 15]. Une alternative consiste à accorder ou à refuser les demandes d'accès des utilisateurs en fonction des attributs de l'utilisateur, des attributs de l'objet et des conditions de l'environnement. Il s'agit du contrôle d'accès basé sur les attributs. Yuan et al. [Yuan, 05] ont proposé le premier modèle ABAC pour sécuriser les architectures SOA (Service Oriented Architectures) et les systèmes distribués. Un modèle ABAC basique comprend deux aspects : "le modèle de politique" qui définit les politiques ABAC et "le modèle d'architecture" qui applique les politiques au contrôle d'accès aux ressources.

[29] https://ranger.apache.org/

3.5.4.1. Modèle de politique ABAC

Le modèle de politique ABAC définit les autorisations en se basant sur un ensemble d'attributs (ou caractéristiques) liés à la sécurité. La logique de contrôle d'accès dans le modèle ABAC consiste à calculer les décisions en fonction des attributs assignés au demandeur d'accès, à la ressource souhaitée et à l'environnement d'accès. La Figure 3.9 illustre une vue d'ensemble du modèle ABAC.

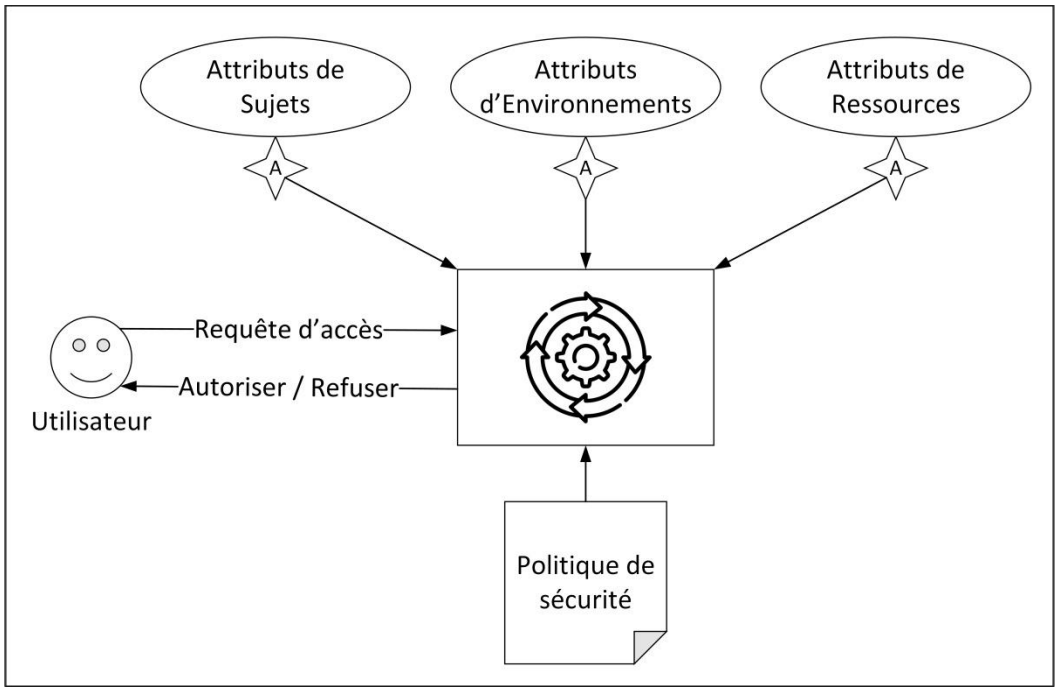

Figure 3.9. Vue d'ensemble du modèle de politique ABAC

Ce modèle définit généralement trois types d'attributs :

- **Attributs de Sujets** : un sujet est une entité qui agit sur une ressource. Chaque sujet a des attributs associés qui définissent son identité et ses caractéristiques. Ces attributs peuvent inclure l'identifiant du sujet, son nom, son organisation, son rôle, etc.
- **Attributs de Ressources** : une ressource est une entité sur laquelle un sujet agit. Comme pour les sujets, les ressources ont des attributs qui peuvent être exploités pour prendre des décisions de contrôle d'accès. Un document texte, par exemple, peut avoir des attributs tels que le chemin, le titre, la date de création, l'auteur, la version, etc.
- **Attributs d'Environnements** : ces attributs décrivent l'environnement ou le contexte opérationnel, technique et situationnel dans lequel se produit l'accès à l'information. Par exemple, des attributs tels que la date et l'heure, la zone géographique, le niveau de sécurité du réseau (par exemple, Internet ou Intranet), etc., ne sont pas associés à un sujet ou à une ressource, mais peuvent néanmoins être pertinents pour l'application d'une politique de contrôle d'accès.

La définition formelle d'un modèle basique de politique ABAC est composée de cinq éléments :

1. S, R et E représentent respectivement des sujets, des ressources et des environnements.
2. SA_k $(1 \leq \mathrm{k} \leq \mathrm{K})$, RA_m $(1 \leq m \leq M)$ et EA_n $(1 \leq n \leq N)$ représentent respectivement les attributs prédéfinis pour les sujets (SA_k), les ressources (RA_m) et les environnements (EA_n).

3. $ATTR(s)$, $ATTR(r)$ et $ATTR(e)$ représentent, respectivement, des relations d'affectation d'attributs pour le sujet s, la ressource r et l'environnement e:

$$ATTR(s) \subseteq SA_1 \times SA_2 \times \ldots \times SA_K$$
$$ATTR(r) \subseteq RA_1 \times RA_2 \times \ldots \times RA_M$$
$$ATTR(e) \subseteq EA_1 \times EA_2 \times \ldots \times EA_N$$

4. Une règle de politique qui décide si un sujet s peut accéder à une ressource r dans un environnement particulier e, est une fonction booléenne des attributs 's', 'r' et 'e' :

$$can\_access\ (s, r, e) \leftarrow f\,(ATTR(s), ATTR(r), ATTR(e))$$

5. Une base de règles de politiques (ou magasin de politiques) peut être constituée d'un certain nombre de règles couvrant des sujets et des ressources au sein d'un domaine de sécurité. Le processus de décision de contrôle d'accès équivaut essentiellement à l'évaluation des règles de politique applicables dans le magasin de politiques.

3.5.4.2. Modèle d'architecture ABAC

Le modèle ABAC permet de créer des règles d'accès sans spécifier de relations individuelles entre chaque sujet et chaque objet. Un sujet se voit attribuer un ensemble d'attributs lors de sa déclaration, comme par exemple, "Alice" est infirmière praticienne au département de cardiologie. Un objet se voit attribuer ses attributs lors de sa création, comme un dossier contenant les fiches médicales des patients cardiaques. Les objets peuvent recevoir leurs attributs soit directement par le créateur, soit par des outils d'analyse automatisés. L'administrateur ou le propriétaire d'un objet crée une règle de contrôle d'accès en utilisant les attributs des sujets et ceux des objets pour régir l'ensemble des capacités autorisées dans un environnement (ou contexte) précis. Par exemple, toutes les infirmières praticiennes du service de cardiologie peuvent, depuis le département de cardiologie, consulter les dossiers médicaux des patients cardiaques. Une architecture d'autorisation typique basée sur des attributs est illustrée dans la Figure 3.10.

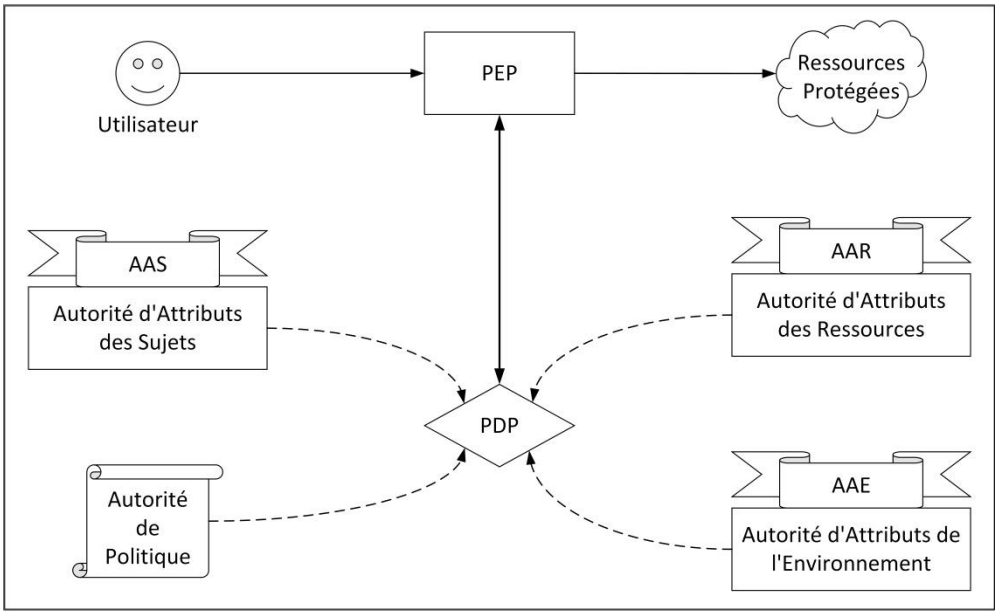

Figure 3.10. Architecture d'Autorisation ABAC (inspiré de [Yuan, 05])

Le schéma de la Figure 3.10 reflète les acteurs logiques impliqués dans un modèle ABAC :

- **Les autorités d'attributs (AA – Attribute Authority)** : elles sont responsables de la création et de la gestion des attributs des sujets, des ressources et de l'environnement. Elles jouent un rôle important dans l'approvisionnement et la découverte des attributs. Une AA peut stocker ou consulter les attributs sur d'autres supports (par exemple, l'AA de Sujets – AAS – peut choisir de récupérer les attributs de l'annuaire LDAP de l'organisation).
- **Le point d'application de la politique (PEP – Policy Enforcement Point)** : ce composant XACML, qu'on verra plus en détail dans la section suivante, est chargé de demander les décisions d'autorisation et de les faire appliquer. Le schéma de la Figure 3.10 représente le PEP comme un point unique, mais il peut être physiquement distribué dans tout le réseau.
- **Le point de décision de politique (PDP – Policy Decision Point)** : ce processus est chargé d'évaluer les politiques applicables et de prendre la décision de permission (autorisation ou refus). Le PDP est par essence un moteur d'exécution de la politique. Lorsqu'une politique fait référence à un attribut d'un sujet, d'une ressource ou de l'environnement qui n'est pas présent dans la demande d'accès, le PDP contacte l'AA appropriée pour récupérer les valeurs de l'attribut.
- **L'autorité de politique (PA – Policy Authority)** : ce composant crée et gère les politiques de contrôle d'accès. Les politiques peuvent consister en des règles de décision, des conditions ou d'autres contraintes pour accéder aux ressources.

3.5.4.3. Application du modèle ABAC dans le contexte Big Data

Les décisions d'accès sous le modèle ABAC peuvent changer en modifiant les valeurs d'attributs, sans devoir modifier les relations sujet/objet définissant les ensembles de règles sous-jacents. Cela fournit une capacité de gestion du contrôle d'accès plus dynamique et limite les exigences de maintenance à long terme pour protéger les objets. En outre, le modèle ABAC peut mettre en œuvre des politiques de contrôle d'accès grâce à un langage de gestion d'attributs, tels que XACML et NGAC, ce qui le rend idéal pour de nombreux environnements distribués ou en évolution rapide [Hu, 15]. En raison de cette flexibilité, ABAC a suscité l'intérêt de l'industrie et de la recherche. Il a été notamment intégré à d'autres approches, comme la norme INCITS pour le contrôle d'accès basé sur les rôles [Kuhn, 10] et est devenu la base d'une gamme croissante de produits. De nombreux travaux de recherche récents utilisent le modèle ABAC pour contrôler l'accès aux infrastructures Big Data, les systèmes IoT et les plateformes du Cloud Computing. Parmi ces travaux, on trouve :

- **Infrastructures Big Data** : Yang [Yang, 20] propose un schéma de contrôle d'accès pour Hadoop par la combinaison du modèle ABAC et du modèle TBDAC (Trust-Based Dynamic Access Control). Le modèle ABAC est utilisé pour rendre l'autorisation de contrôle d'accès plus raisonnable en formulant un ensemble complet de politiques d'attributs et en tenant compte des facteurs d'environnement. Gupta et al. [Gupta, 18] proposent des extensions aux capacités d'autorisation offertes par Hadoop. Le modèle proposé, appelé HeABAC, est un modèle de contrôle d'accès basé sur des attributs à granularité fine répondant aux besoins de sécurité et de confidentialité de l'écosystème Hadoop multi-locataire.
- **Systèmes IoT** : Gupta et al. [Gupta, 20] proposent un système formel de contrôle d'accès basé sur les attributs pour résoudre les problèmes de contrôle d'accès dans l'écosystème de l'Internet Industriel des Véhicules IIoV (Industrial Internet-of-Vehicles). Le modèle proposé introduit la notion de groupes attribués à diverses entités intelligentes basées sur différents attributs. Ce modèle prend en compte les préférences de confidentialité individualisées ainsi que les politiques

à l'échelle du système pour accepter ou rejeter les notifications, les alertes et les publicités des différentes entités intelligentes participantes. Dans la même catégorie, Zhang et al. [Zhang, 18-b] présentent PASH (Privacy-Aware Smart-Health), un système de contrôle d'accès intelligent préservant la vie privée des individus. Ce système est basé sur un schéma de chiffrement dans lequel les politiques d'accès sont partiellement masquées et les valeurs d'attributs sont chiffrées (seuls les noms sont révélés).

- **Cloud Computing** : pour construire un système de contrôle d'accès basé sur des attributs multi-autorités sécurisé et rentable pour le partage de données dans les systèmes de stockage en Cloud, Wei et al. [Wei, 16] proposent un schéma de chiffrement prenant en considération la révocation évolutive des utilisateurs et la mise à jour des données chiffrées. Dans cette même thématique, Liu et al. [Liu, 13] proposent un schéma de contrôle d'accès basé sur des attributs hiérarchiques et une structure hiérarchique des autorités.

Le modèle ABAC a le potentiel d'améliorer considérablement le contrôle d'accès dans les environnements Big Data. Toutefois, une définition consensuelle des attributs est nécessaire ; cela peut rendre complexe la protection de données dans des applications Big Data modernes telles que les environnements intelligents ou encore les réseaux sociaux. En effet, l'implémentation du modèle ABAC peut nécessiter une infrastructure complexe pour la gestion des attributs et des politiques ainsi qu'un éventail complet de fonctions qui prend en charge les décisions d'accès et l'application des politiques.

3.6. Langages de contrôle d'accès

Dans le déploiement et l'application des modèles et des technologies de contrôle d'accès, plusieurs langages de contrôle d'accès ont été développés pour mettre en œuvre efficacement l'accès des utilisateurs et la gestion des autorisations. Un langage de contrôle d'accès est le pont entre la théorie et la pratique du contrôle d'accès [Cai, 18]. Voici quatre langages de contrôle d'accès courants :

- Le langage **SAML** (Security Assertion Markup Language) [Cantor, 05] : ce langage, basé sur la norme XML, peut être utilisé pour implémenter l'échange des données d'authentification et d'autorisation entre différents domaines de sécurité. SAML permet de définir la déclaration des attributs et des autorisations ainsi que la gestion du processus d'authentification.
- Le langage **SPML** (Service Provisioning Markup Language) [Cole, 05] : ce langage, également développé sur la base de la norme XML, est principalement utilisé pour créer des demandes de service pour les comptes des utilisateurs et les demandes liées à la gestion des services. Ce langage facilite la configuration des exigences de sécurité et d'audit et simplifie l'implémentation de l'interopérabilité entre différents systèmes de configuration.
- Le langage **XACML** (eXtensible Access Control Markup Language) [Rissanen, 17] : XACML est une spécification basée sur XML qui définit un langage et un protocole pour exprimer les règles de sécurité et administrer les politiques de contrôle d'accès.
- Le langage **NGAC** (Next Generation Access Control) [Ferraiolo, 18] : NGAC définit le contrôle d'accès en termes d'abstractions de données et de fonctions basées sur des attributs associés aux utilisateurs, aux processus et aux objets.

Dans le reste de cette section, nous présentons XACML et NGAC qui sont les deux standards courants qui fournissent des moyens normalisés pour exprimer et appliquer des politiques de contrôle

d'accès. Bien qu'ils supportent tout type de modèles de contrôle d'accès, ces deux langages sont nativement adaptés pour les modèles de contrôle d'accès basés sur les attributs.

3.6.1. XACML

Le langage XACML, disponible depuis 2003, a été conçu pour exprimer les politiques de sécurité ainsi que les demandes d'accès et les réponses nécessaires pour interroger un système de sécurité. XACML a été développé dans le but de séparer l'expression de politiques et la prise de décision. Sa dernière version 3.0 Errata 01 a été publiée en 2017 [Rissanen, 17]. XACML fournit un protocole de type client-serveur dont l'architecture sous-jacente simplifiée est illustrée dans la Figure 3.11.

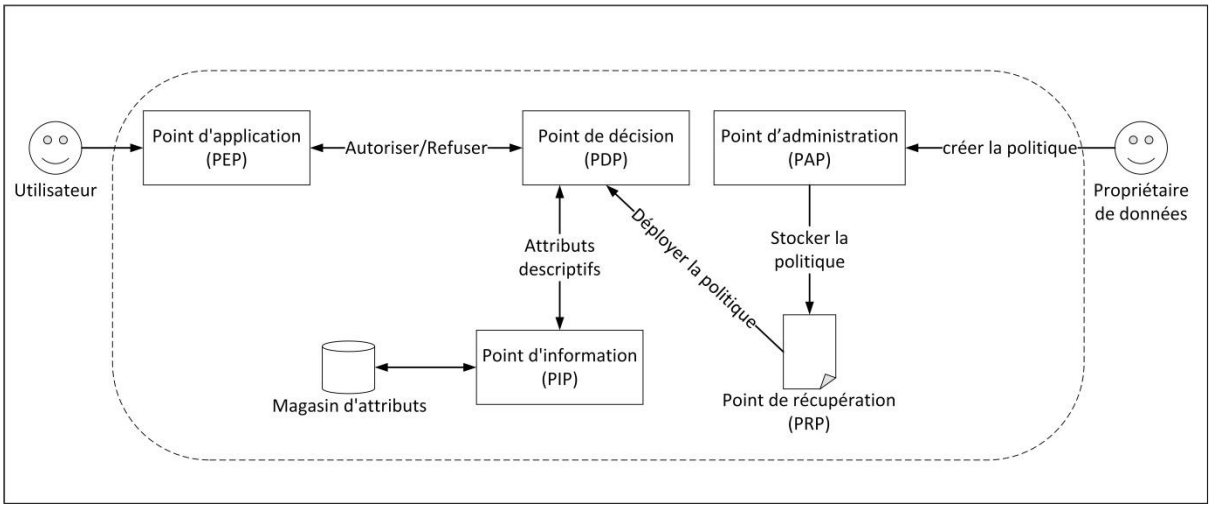

Figure 3.11. Architecture fonctionnelle simplifiée de XACML

L'architecture type de XACML est composée de cinq éléments : le point d'application de la politique **PEP** (Policy Enforcement Point), le point de décision de la politique **PDP** (Policy Decision Point), le point d'information sur la politique **PIP** (Policy Information Point), le point d'administration de la politique **PAP** (Policy Administration Point) et le point de récupération de la politique **PRP** (Policy Retrieval Point).

Dans le diagramme de flux de données XACML, un utilisateur (le sujet) fait appel au PEP, le point d'application de la politique, afin de décider si une requête d'accès est autorisée. En pratique, ce PEP peut être vu comme un Proxy ; son rôle est d'interrompre ou d'autoriser la requête reçue en fonction du résultat de l'échange avec le PDP. Ce dernier étant le moteur de gestion des autorisations qui évalue la requête reçue du PEP et lui renvoie une décision (Autoriser/Refuser). Pour calculer les décisions, le PDP peut avoir besoin d'interroger un point d'information, le PIP, pour recueillir des attributs descriptifs sur l'utilisateur, les objets souhaités, l'environnement ou tout autre attribut manquant dans la demande. Ces attributs peuvent être récupérés en se connectant à des magasins d'attributs comme LDAP ou une base de données externe. Le PAP est le point d'administration des politiques ; c'est au niveau de ce point que les politiques de contrôle d'accès sont gérées par les propriétaires des données ou l'administrateur du

système. Enfin, le PRP est le point de stockage des politiques ; il s'agit d'une base de données dans laquelle le PDP peut accéder aux politiques.

Les politiques dans XACML sont structurées en tant qu'ensembles de politiques stockées dans le PRP. Chaque ensemble de politiques est composé de plusieurs politiques et éventuellement d'autres sous-ensembles de politiques. Une politique est composée d'un ensemble de règles de sécurité. XACML inclut la notion de "cible" (ou Target, en anglais) dans les règles de sécurité. Une cible définit une condition booléenne simple qui, si elle est satisfaite par les attributs, établit la nécessité d'une évaluation ultérieure par le PDP. Si aucune cible ne correspond à la demande (i.e. aucun attribut ne correspond à la cible), la décision calculée par le PDP est "Non-Applicable" ; le PEP refusera alors l'accès pour absence d'attribut.

Une règle comprend, en plus de la cible, une série de conditions booléennes qui, si elles sont évaluées à "vraies", ont un effet sur la décision d'autorisation ou de refus. Les conditions d'une règle sont généralement plus complexes et peuvent inclure des fonctions de comparaison sur les valeurs d'attributs telles que "supérieur ou égal à", "appartenir à une liste", "égal à une chaîne", etc. Les conditions peuvent être utilisées pour exprimer les relations de contrôle d'accès (par exemple, un médecin ne peut consulter que les dossiers médicaux de ses patients).

Étant donné qu'une politique peut contenir plusieurs règles et qu'un ensemble de politiques peut contenir plusieurs politiques et plusieurs sous-ensembles de politiques, chaque règle, politique ou ensemble de politiques peut évaluer des décisions différentes (autorisation, refus ou non-applicable). XACML fournit un moyen de réconciliation grâce à une collection d'algorithmes de combinaison. Chaque algorithme représente une manière différente de combiner plusieurs décisions locales en une seule décision globale. Il existe plusieurs algorithmes de combinaison, dont les suivants :

- Surcharge de refus : si une décision est évaluée à "Refuser", ou si aucune décision n'est évaluée à "Autoriser", alors le résultat est "Refuser". Si toutes les décisions sont évaluées à "Autoriser", le résultat est "Autoriser".
- Surcharge d'autorisation : si une décision est évaluée à "Autoriser", alors le résultat est "Autoriser", sinon le résultat est "Refuser".

Les algorithmes de combinaison sont appliqués aux règles dans une politique et aux politiques dans un ensemble de politiques pour arriver à une décision finale du PDP. La combinaison d'algorithmes peut être utilisée pour élaborer des politiques plus complexes.

XACML inclut également les concepts d'expressions d'obligation et de conseil. Une obligation spécifiée dans une règle, une politique (ou un ensemble de politiques), est une directive du PDP au PEP sur ce qui doit être effectué avant ou après l'approbation ou le refus d'une demande d'accès. Le conseil est similaire à une obligation, sauf que le conseil peut être ignoré par le PEP.

3.6.2. NGAC

NGAC est une norme émergente conçue pour exprimer et appliquer des politiques de contrôle d'accès, à travers la configuration d'un ensemble d'éléments appelés "relations". Avant de présenter l'architecture NGAC, nous allons présenter certaines notions élémentaires définies par NGAC.

NGAC utilise les termes utilisateur, opération et objet avec des significations similaires aux termes sujet, action et ressource de XACML. Pour pouvoir effectuer une opération, un ou plusieurs droits d'accès sont nécessaires. NGAC comprend également des processus qui sont des entités indépendantes des utilisateurs mais qui peuvent être liées ; les processus par lesquels un utilisateur tente d'accéder à des objets prennent les mêmes attributs que l'utilisateur appelant. Les données de contrôle d'accès de NGAC sont constituées d'un ensemble d'éléments basiques (i.e. les utilisateurs, les objets et les droits d'accès des utilisateurs pour effectuer des opérations), des conteneurs (contenant les attributs d'utilisateurs, les attributs d'objets et les classes de politiques) et des relations qui permettent d'exprimer des politiques (l'équivalent des règles dans XACML). Quatre types de relations existent :

- les affectations qui permettent de définir des règles d'appartenance dans les conteneurs,
- les associations qui permettent de spécifier des privilèges,
- les interdictions qui permettent de dériver des exceptions de privilège,
- et enfin, les obligations qui permettent de modifier dynamiquement l'état d'accès.

Nous allons présenter tous ces éléments dans le reste de cette section.

3.6.2.1. Les données de contrôle d'accès de NGAC

3.6.2.1.1. Conteneurs

Les conteneurs jouent un rôle déterminant dans la formulation et l'administration des politiques et des attributs. Les conteneurs d'utilisateurs (ou les attributs d'utilisateurs) peuvent représenter des rôles, des affiliations ou toute autre caractéristique commune à des utilisateurs. Les conteneurs d'objets (ou les attributs d'objets) caractérisent les données et les autres ressources. Dans NGAC, un objet fait référence explicite vers l'attribut d'objet à travers son nom ; autrement dit, un objet est lui-même un attribut d'objet. Les conteneurs d'objets contiennent donc les objets et peuvent en plus représenter des objets composés, tels que des dossiers, des colonnes de tableau ou des lignes (pour répondre à des exigences plus fines). Enfin, les conteneurs de classes de politiques sont utilisés pour regrouper et caractériser des collections de politiques ou des services de données. Chaque utilisateur, attribut d'utilisateur et attribut d'objet doit appartenir à au moins une classe de politiques. Les classes de politiques peuvent être mutuellement exclusives ou se chevaucher à divers degrés pour répondre à une gamme d'exigences de politique.

3.6.2.1.2. Relations

Quatre types de relations existent dans NGAC : les affectations, les associations, les interdictions et les obligations. Dans cette section, nous allons présenter chacune de ces relations.

3.6.2.1.2.1. Affectations et Associations

NGAC utilise un tuple (x, y) pour spécifier l'affectation de l'élément x à l'élément y. Dans ce chapitre, nous utilisons la notation x → y pour désigner que l'élément x est affecté à l'élément y. La relation d'affectation implique toujours l'appartenance (x appartient à y, ou bien, x est contenu dans y). L'ensemble des entités utilisées dans les affectations comprend les utilisateurs, les attributs d'utilisateurs, les attributs d'objets et les classes de politiques.

Les droits d'accès pour effectuer des opérations sont acquis par le biais d'associations. Une association est un triplet, désigné par *ua--ars--at*, où *ua* est un attribut d'utilisateur, *ars* (access rights set)

est un ensemble de droits d'accès, et *at* est un attribut qui peut désigner soit un attribut d'utilisateur soit un attribut d'objet. L'attribut *at* dans une association est utilisé comme référent pour lui-même et les éléments de politique contenus par l'attribut. De même, le premier terme de l'association, l'attribut *ua*, est traité comme un référent pour les utilisateurs contenus dans *ua*. La signification de l'association *ua--ars--at* est que les utilisateurs contenus dans *ua* peuvent exécuter les droits d'accès dans *ars* sur les éléments de politique référencés par *at*. Un exemple des relations d'affectations et d'associations est illustré dans la Figure 3.12.

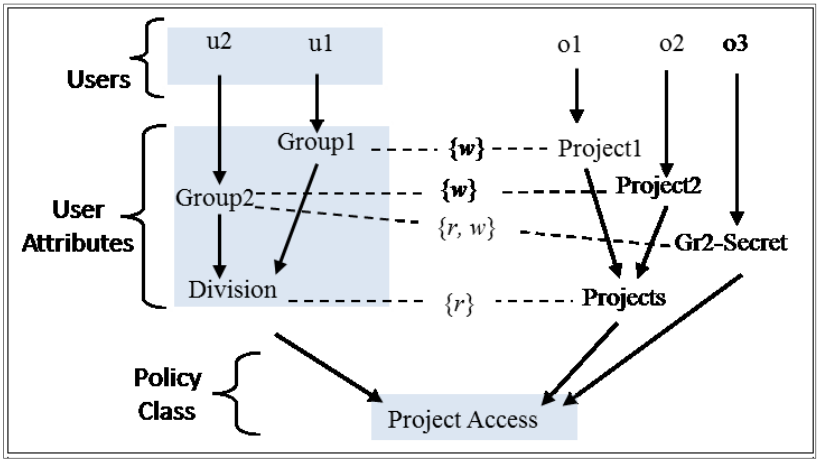

Figure 3.12. NGAC - Exemple des relations d'affectations et d'associations [Ferraiolo, 16]

Dans l'exemple de la Figure 3.12 nous avons :

- Deux utilisateurs (*u1* et *u2*) et trois attributs d'utilisateurs, ou conteneurs d'utilisateurs, (*Group1*, *Group2* et *Division*). Les relations d'affectation sont représentées par des flèches dirigées (*u1* et *u2* sont contenus respectivement par *Group1* et *Group2* qui sont eux-mêmes contenus par l'attribut d'utilisateur *Division*.
- Trois objets (*o1*, *o2* et *o3*) qui font référence à des attributs d'objets représentés par leurs noms respectifs o1, o2 et o3 et quatre conteneurs d'objets (*Project1*, *Project2*, *Projects* et *GR2-Secrect*). Les objets *o1* et *o2* sont respectivement contenus par *Project1* et *Project2* qui sont eux-mêmes contenus par l'attribut d'objet *Projects*. L'objet *o3* est contenu par l'attribut d'objet *GR2-Secrect*.
- Une classe de politique, appelée *Project Access*, qui contient l'attribut d'utilisateur *Division* et les attributs d'objets *Projects* et *GR2-Secrect*.
- Les associations sont représentées par des lignes pointillées. Par exemple, les utilisateurs du *Group2* ont le droit d'écriture *{w}* sur tous les objets contenus dans *Project2* et le droit de lecture/écriture *{r, w}* sur tous les objets contenus dans *GR2-Secrect*.

Un privilège (ou relation dérivée), de la forme *(u, ar, e)*, signifie que l'utilisateur *u* est autorisé à exécuter le droit d'accès *ar* sur l'élément *e*, où e peut représenter un utilisateur, un attribut d'utilisateur ou un attribut d'objet. L'existence d'un privilège est une exigence, mais insuffisante, pour calculer une décision d'accès. En effet, la décision est calculée en fonction des privilèges et des interdictions que nous verrons dans la section suivante. Voici la liste des privilèges du graphe de la Figure 3.12 : (u1, w, o1), (u1, r, o1), (u1, r, o2), (u2, w, o2), (u2, r, o3), (u2, w, o3), (u2, r, o1), (u2, r, o2).

Une opération peut être administrative ou non-administrative. La différence c'est qu'une opération administrative porte sur des objets nécessitant des droits d'accès de lecture et/ou d'écriture tandis qu'une opération non-administrative porte sur des éléments de politiques ou des relations (changement d'une affectation, d'une association…). Qu'elles soient administratives ou non-administratives, les opérations sont exprimées grâces à des associations.

3.6.2.1.2.2. Interdictions et Obligations

En plus des affectations et des associations, NGAC comprend des relations d'interdiction qui spécifient des exceptions de privilège. Trois types de relations d'interdiction existent dans NGAC : refus d'utilisateur (user deny), refus d'attribut d'utilisateur (user attribute deny) et refus de processus (process deny). Admettons que les notations *user_deny(u, ars, pe), user_attribute_deny(ua, ars, pe)* et *process_deny(p, ars, pe)* désignent respectivement un refus d'utilisateur, un refus d'attribut d'utilisateur et un refus de processus où *u* est un utilisateur, *ua* est un attribut d'utilisateur, *p* est un processus, *ars* est un ensemble de droits d'accès et *pe* est un élément de politique utilisé comme référent pour lui-même et pour les éléments de politique qu'il contient. Ces relations signifient que l'utilisateur *u*, les utilisateurs dans *ua* et le processus *p* ne peuvent pas exécuter des droits d'accès dans *ars* sur des éléments de politique dans *pe*. Les relations de refus d'utilisateur et les relations de refus d'attribut d'utilisateur peuvent être créées directement par un administrateur ou dynamiquement en conséquence d'une obligation. Lorsqu'elles sont créées via une obligation, les relations de refus d'utilisateur et d'attribut d'utilisateur peuvent prendre des conditions de politique dynamiques. De telles conditions peuvent, par exemple, prendre en charge les politiques de séparation des tâches. Enfin, les relations de refus de processus sont créées exclusivement à l'aide d'obligations. Leur utilisation principale est l'application de conditions d'affectation (par exemple, si un processus lit des données de la classe "Top Secret" alors ce processus n'a pas le droit d'écrire sur un objet qui ne figure pas dans la classe "Top Secret").

De plus, le composant d'élément de politique de chaque relation d'interdiction peut être spécifié comme son complément, désigné par ¬. La signification respective des notations *user_deny(u, ars, ¬pe), user_attribute_deny(ua, ars, ¬pe) et process_deny(p, ars, ¬pe)* est que l'utilisateur *u*, tout utilisateur affecté à *ua* et le processus *p* ne peuvent pas exécuter les droits d'accès dans *ars* sur des éléments de stratégie qui ne sont pas dans *pe*.

Les obligations consistent en une paire *(ep, r)* où *ep* est un modèle d'événement (event pattern, en anglais) et *r* (pour response, en anglais) est une séquence d'opérations administratives, appelée réponse. Le modèle d'événement *ep* spécifie des conditions qui, si elles correspondent au contexte entourant l'exécution réussie d'un processus d'une opération sur un événement, entraînent l'exécution immédiate des opérations administratives de la réponse *r* associée. Les obligations sont généralement exprimées sous la forme : « *when ep do r* ». Si on reprend l'exemple de la Figure 3.12, même si cette politique suggère que seuls les utilisateurs du Group2 peuvent lire *Gr2-Secrets*, les données de *Gr2-Secrets* peuvent en effet être divulguées aux utilisateurs du *Group1*. Plus précisément, *u2* (ou l'un de ses processus) peut lire *o3*, puis écrire son contenu dans *o2*, fournissant ainsi à *u1* la capacité de lire le contenu de *o3* (à travers l'association *Division---{r}---Projects*). Une telle fuite peut être évitée avec l'obligation suivante:

When any process p performs (r, o) where o → Gr2-Secret do create process-deny(p, {w}, ¬Gr2-Secret)

¬*Gr2-Secret* signifie « n'est pas » dans *Gr2-Secret.* L'effet de cette obligation empêchera un processus (et son utilisateur) de lire un objet dans *Gr2-Secret* et d'écrire ensuite son contenu dans un objet dans un conteneur différent.

3.6.2.1.3. Fonctions de décisions d'accès

La fonction de décision d'accès contrôle les accès en termes de processus. L'utilisateur au nom duquel le processus fonctionne doit détenir une autorité suffisante sur les éléments de politique impliqués. La fonction *process_user(p)* désigne l'utilisateur associé au processus *p*. Les demandes d'accès sont de la forme *(p, op, argseq)*, où *p* est un processus, *op* est une opération et *argseq* est une séquence d'un ou plusieurs arguments compatibles avec la portée de l'opération. La fonction de décision d'accès pour déterminer si une demande d'accès peut être accordée nécessite une mise en correspondance d'une paire d'opérations et de séquences d'arguments *(op, argseq)* à un ensemble de droits d'accès et de paires d'éléments de politique *{(ar, pe)}*. Lors de la détermination d'accorder ou de refuser une demande d'accès, la fonction de décision d'accès prend en compte tous les privilèges et les restrictions (refus) qui s'appliquent à un utilisateur et à ses processus, qui sont dérivés des associations et des interdictions.

3.6.2.1.4. Délégation des capacités administratives

L'initialisation du système est réalisée par un "super-utilisateur" doté de capacités pour effectuer toutes les opérations administratives sur toutes les données du contrôle d'accès. L'état initial consiste en une configuration NGAC avec des éléments de données, des attributs et des relations vides. Le super-utilisateur peut créer directement des capacités administratives ou, plus concrètement, créer des administrateurs et leur déléguer des capacités pour créer et supprimer des privilèges administratifs. La délégation et l'annulation des capacités administratives sont obtenues par la création et la suppression d'associations.

3.6.2.2. Architecture NGAC

Le flux de données du standard NGAC est illustré dans la Figure 3.13. L'architecture reprend certains éléments de XACML qui ont le même rôle, à quelques différences près, et en ajoute d'autres éléments. Le PEP est le point d'application de la politique. Le RAP (Resource Access Point) est le point d'accès aux ressources. Le PDP est le point de décision. Le PAP est le point d'administration de la politique. Le PIP est le point d'information de la politique. Et enfin, l'EPP (Event Processing Point) est un point optionnel dédié au traitement des évènements.

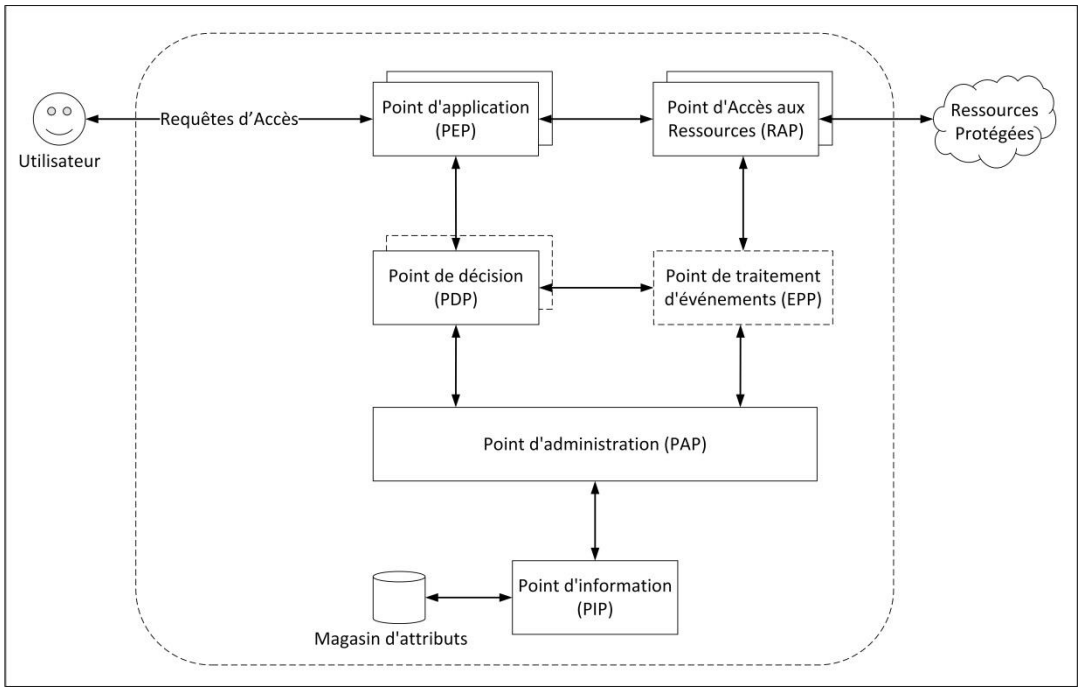

Figure 3.13. Architecture fonctionnelle simplifiée de NGAC

Le PEP intercepte les demandes d'accès. Une demande d'accès comprend l'identifiant de l'utilisateur (qui peut être une personne, un processus ou une application), l'opération désirée (lecture, écriture, …), et une séquence d'un ou de plusieurs opérandes mandatés par l'opération qui se rapportent à une ressource ou à une relation de contrôle d'accès. Pour déterminer s'il faut accorder ou refuser, le PEP soumet la demande à un PDP (il peut y avoir plusieurs PDP) qui calcule la décision en fonction de la configuration des éléments de données et des relations stockées dans le PIP, via le PAP. Le PDP renvoie la décision d'autorisation ou de refus au PEP. Si l'accès est accordé et que l'opération a été en lecture/écriture, le PDP renvoie également l'emplacement physique où réside le contenu de la ressource désirée. Le PEP envoie ensuite une commande au RAP approprié pour exécuter l'opération sur les ressources. Si la demande porte sur une opération administrative et que la décision a été accordée, le PDP émet une commande au PAP pour l'exécution de l'opération sur l'élément de données ou la relation stockée dans le PIP. Le RAP et le PAP renvoient les statuts d'exécution ; si le statut renvoyé par le RAP ou le PAP est "réussi", le PEP soumet le contexte de l'accès à l'EPP (s'il existe). Si le contexte correspond à un modèle d'événement d'une obligation, l'EPP exécute automatiquement les opérations administratives de cette obligation, modifiant potentiellement l'état d'accès.

3.7. Contrôle d'accès en Big Data

De nombreux modèles de contrôle d'accès ont été proposés dans la littérature ; chacun traite une problématique spécifique. Dans cette section, nous allons présenter une panoplie de modèles de contrôle d'accès répartis selon quatre domaines en étroite liaison avec la Big Data :

- Les environnements distribués et collaboratifs
- Les environnements de stockage sur le Cloud et le Cloud Computing
- Les environnements intelligents et les systèmes IoT

- La plateforme Hadoop et son écosystème

3.7.1. Contrôle d'accès pour les environnements distribués et collaboratifs

Le modèle DTMAC (Dynamic TeaM Access Control) [Benhadj Djilali, 19] est une extension des modèles RBAC et TMAC (TeaM Based Access Control). L'idée de base est de créer des équipes collaboratives de façon dynamique à chaque fois que l'on souhaite exercer une activité collaborative. DTMAC est applicable dans les environnements IoT. Une équipe collaborative, composée d'utilisateurs et de capteurs, a une durée de vie limitée qui se termine à la fin de l'activité collaborative. Les auteurs distinguent entre deux rôles : Maître et Esclave. Un sujet ayant le rôle Maître a la responsabilité de gérer une équipe pour réaliser une activité collaborative (de la création de l'équipe, à l'affectation des rôles aux esclaves, jusqu'à la suppression de l'équipe). Les sujets ayant le rôle Esclave, quant à eux, peuvent appartenir simultanément à plusieurs équipes collaboratives. Par contre, les données d'un capteur ne sont accessibles que par les membres de l'équipe par laquelle il est géré (un capteur appartient à une et une seule équipe). L'activation des rôles se fait à travers des sessions en suivant le même principe que dans RBAC. DTMAC repose sur le noyau RBAC, ce qui veut dire qu'il ne gère pas les contraintes de séparation de fonctions ni la hiérarchie des rôles. Par ailleurs, la collaboration implique que plusieurs utilisateurs travaillent ensemble pour réaliser une activité sans forcément qu'ils appartiennent tous à la même équipe. Selon le modèle DTMAC, à chaque fois qu'un utilisateur externe de l'équipe doit accéder à une donnée générée par un capteur, le Maître doit l'ajouter à l'équipe pour réaliser son activité. Cela rend l'administration du modèle très complexe notamment dans un environnement distribué tel que l'IoT.

Certains travaux s'intéressent aux opérations inter-domaines pour réaliser des activités collaboratives. Par "domaine" on désigne un environnement composé d'un ensemble d'utilisateurs agissant sur des ressources (ou objets) selon une politique de sécurité propre. Parmi ces travaux on trouve [Lv, 16] dans lequel les auteurs introduisent deux types d'hiérarchie par domaine : une hiérarchie pour les utilisateurs représentés par leurs rôles et une hiérarchie pour les ressources. Chaque utilisateur est assigné à un rôle (dans la hiérarchie des rôles) qui lui autorise certaines permissions pour manipuler des ressources (dans la hiérarchie des ressources). Cette permission est récursive : grâce à son rôle, un utilisateur accède non seulement à l'objet correspondant dans l'arbre des ressources mais aussi à tous les objets enfants de cet objet en plus des tous les objets auxquels les rôles enfants peuvent accéder. Dans ce modèle, les auteurs introduisent deux notions : la "liaison de rôle" d'un domaine vers un autre et le "transfert de liaison". Admettons qu'un rôle R^A de l'arbre T^A du domaine A est lié à un rôle R^B de l'arbre T^B d'un autre domaine B : si le transfert de liaison est activé, alors le rôle R^A devient « supérieur » à tous les enfants du rôle R^B ; sinon, si le transfert liaison n'est pas activé, alors le rôle R^A ne peut accéder qu'aux objets accessibles directement par le rôle R^B mais pas à ceux accessibles par les enfants de ce dernier. Les auteurs considèrent ainsi trois politiques d'accès possibles :

- Politique d'accès par défaut : le transfert est activé pour toutes les relations de liaison.
- Politique d'accès directe : le transfert est désactivé pour toutes les relations de liaison.
- Politique d'accès partielle : c'est l'administrateur qui décide pour quelles relations de liaison le transfert est activé et pour quelles relations il est désactivé.

Le modèle proposé par Lv et al. permet d'assurer des opérations inter-domaines mais reste très complexe à gérer notamment quand la politique d'accès est partielle (ce qui est généralement le cas). Il

faudra aussi gérer manuellement les relations de liaison pour tous les rôles et tous les objets ce qui peut entrainer des erreurs humaines. En outre, ce modèle est statique et ne gère pas les aspects contextuels d'accès (environnement, exceptions, …).

Dans cette même thématique on trouve le modèle domRBAC proposé par Gouglidis et Mavridis [Gouglidis, 12] qui reprend les concepts de base du standard RBAC et ajoute un nouveau concept qui est le "conteneur" pour gérer certains aspects contextuels. Ce dernier est un élément abstrait qui incorpore des facteurs de décision employés par la fonction de décision d'accès. Un conteneur gère des attributs liés à l'environnement (des contraintes de temps, des informations spatiales, etc.) et d'autres attributs liés au niveau d'utilisation qui peuvent être utilisés pour limiter l'utilisation des ressources partagées. De plus, domRBAC introduit des contraintes de cardinalité des rôles qui peuvent être appliquées au processus d'attribution et d'activation des rôles. Cela signifie que le nombre de rôles pouvant être attribués et/ou activés par les utilisateurs d'un système peut être géré afin de satisfaire les exigences posées par les administrateurs du système et celles des propriétaires des ressources. Le contrôle d'accès inter-domaines est la différence fondamentale entre RBAC et domRBAC. C'est une fonctionnalité qui régit les inter-opérations entre les domaines grâce à l'héritage des rôles inter-domaines. Cependant, la hiérarchisation des rôles intra-domaines et inter-domaines avec les différentes contraintes adoptées par le modèle domRBAC rendent son administration très complexe notamment dans un environnement Big Data où l'on trouve plusieurs équipes, chacune gère plusieurs domaines. En outre, les contraintes de cardinalité, notamment celles qui sont dynamiques, peuvent bloquer temporairement l'accès à certains rôles pour effectuer certaines opérations inter-domaines qui peuvent être urgentes, d'où la violation de la propriété de disponibilité ; une des principaux objectifs d'un système de sécurité.

3.7.2. Contrôle d'accès pour les systèmes de stockage sur le Cloud et le Cloud Computing

Le stockage sur le Cloud est l'un des principaux services fournis par le Cloud Computing. Il permet aux propriétaires de données d'héberger leurs données dans des serveurs externes. Cependant, ce nouveau paradigme remet en question les approches de contrôle d'accès traditionnel et, par conséquent, de nombreuses techniques bien adaptées au contrôle d'accès aux données externalisées sur le Cloud sont apparues. Parmi ces techniques, on trouve le chiffrement basé sur les attributs ABE (Attribute-Based Encryption). Ces techniques ont été largement abordées dans la littérature [Bethencourt, 07], [Goyal, 06], [Lewko, 10], [Lewko, 11], [Waters, 11], [Wei, 16], [Xiao, 15], [Yang, 14], [Zhang, 18-b]. Le chiffrement basé sur les attributs, introduit pour la première fois par Sahai et Waters [Sahai, 05], consiste à ce qu'un propriétaire de données désirant externaliser ses données sur le Cloud commence par définir une politique d'accès basée sur les attributs et chiffre ensuite ses données dans le cadre de cette politique. Dans l'exemple de la Figure 3.14, chaque utilisateur est décrit par un ensemble d'attributs et détient une Clé Secrète (CS) émise par une autorité d'attributs.

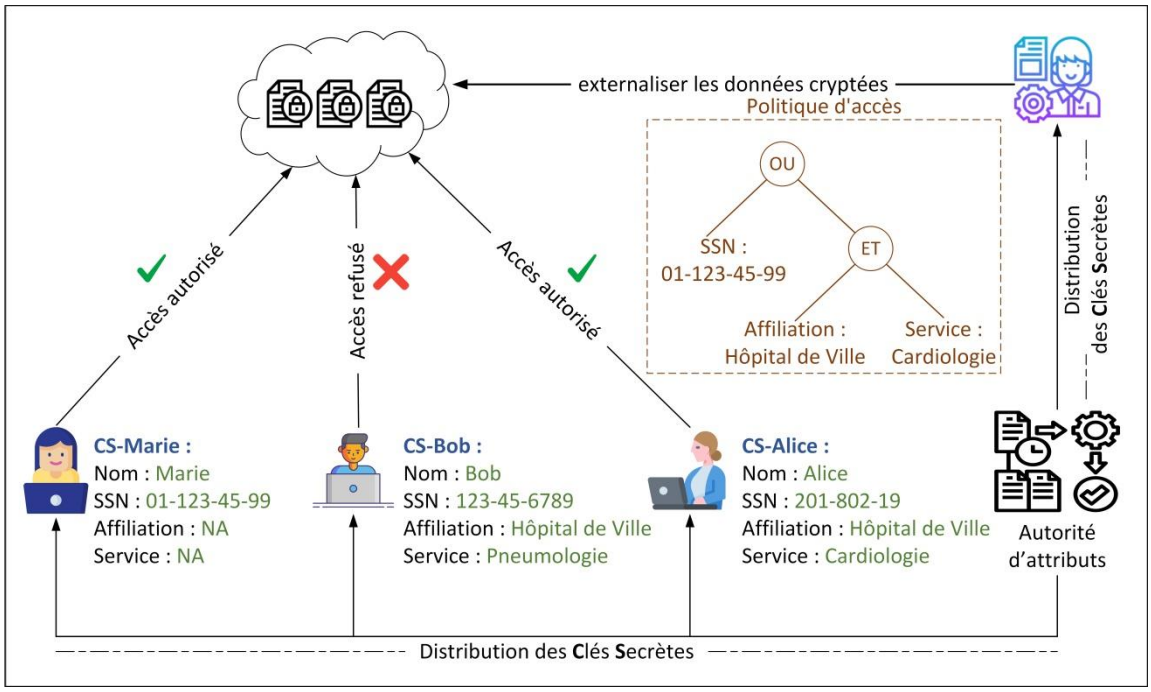

Figure 3.14. Chiffrement basé sur les attributs (inspiré de [Wei, 16] et [Zhang, 18-b])

Selon la politique d'accès adoptée dans l'exemple de la Figure 3.14, Marie, qui a un numéro de sécurité sociale SSN = 01-123-45-99, peut déchiffrer les données par sa clé CS-Marie. Alice, comme elle est Cardiologue à l'Hôpital de Ville, peut également déchiffrer les données par sa clé CS-Alice. Bob, quant à lui, même s'il travaille à l'Hôpital de Ville, il ne peut pas déchiffrer les données parce qu'il n'est pas Cardiologue. Ainsi, seuls les utilisateurs dont les attributs satisfont à la politique d'accès peuvent déchiffrer les données externalisées. Cela empêche les utilisateurs non-autorisés, y compris les fournisseurs des serveurs Cloud, de lire les données externalisées.

En pratique, les systèmes de sécurité sur le Cloud adoptent souvent plusieurs autorités d'attributs plutôt qu'une seule. Ces autorités d'attributs sont en général indépendantes les unes des autres et chacune gère un domaine particulier ; chaque autorité génère une partie de la clé secrète pour chaque utilisateur, en fonction des attributs de son domaine. Le rassemblement de l'ensemble des parties constitue la clé secrète complète d'un utilisateur.

Goyal et al. [Goyal, 06] ont défini deux catégories pour les politiques de chiffrement basé sur les attributs : les politiques d'accès basées sur la clé, KP-APE (Key-Policy Attribute-Based Encryption) et les politiques d'accès basées sur le message chiffré, CP-ABE (Ciphertext-Policy Attribute-Based Encryption). Dans le cadre de KP-ABE, les clés secrètes sont associées à des politiques d'accès et les données chiffrées sont décrites par des attributs. En revanche, CP-ABE consiste à définir une politique d'accès dans le message chiffré lui-même ; un utilisateur ne peut déchiffrer un message que si les attributs qui lui sont assignés correspondent à la politique d'accès définie pour le message. Il s'agit donc de deux méthodes différentes pour atteindre le même objectif qui est le chiffrement basé sur les attributs. KP-ABE est plutôt adaptée aux applications de requête (appel de web service, requête dans une base de données, etc.) tandis que CP-ABE convient parfaitement aux applications de contrôle d'accès aux données confidentielles.

Pour protéger les données externalisées sur le Cloud, le chiffrement et le contrôle d'accès basés sur les attributs doivent relever un ensemble de défis parmi lesquels on cite :

1. Le changement des politiques d'accès par les propriétaires de données. Il est souvent nécessaire que les particuliers et les entreprises publiant des données sur le Cloud changent les règles d'accès à leurs données en ajoutant, modifiant ou même supprimant certains attributs.
2. Le changement des droits d'un utilisateur. Cela comprend deux catégories : le changement des attributs assignés à l'utilisateur (ajout ou suppression d'attributs) et la révocation d'un utilisateur.
3. Le masquage des politiques. Une politique d'accès exprimée sous forme de texte en clair peut révéler des informations sensibles à travers les valeurs des attributs. Si on prend l'exemple de la Figure 3.14, la politique d'accès spécifie que les données chiffrées ne peuvent être consultées que par un Cardiologue de l'Hôpital de Ville ou par un patient avec le numéro de sécurité sociale 01-123-45-99. Si cette politique n'est pas masquée, de toute évidence, tout utilisateur ayant accès au serveur Cloud peut en déduire qu'un utilisateur ayant le numéro de sécurité sociale 01-123-45-99 souffre probablement d'un problème cardiaque.

Relever ces défis pose un grand problème quand il s'agit d'un environnement Big Data (de très grandes quantités de données, de nombreux utilisateurs, un grand univers d'attributs, etc.). Dans la suite de cette section, nous présentons quelques modèles de contrôle d'accès qui traitent ces problématiques.

Dans [Wei, 16], les auteurs proposent un cadre de contrôle d'accès basé sur des attributs, multi-autorités, pour les systèmes de stockage de données sur le Cloud. Leur approche implique cinq entités :

- Tiers : cette entité est chargée de produire le paramètre public global du système, qui est partagé entre toutes les autorités, les propriétaires et les utilisateurs du système.
- Autorités d'Attribut : chaque autorité d'attributs gère son propre univers d'attributs et définit son propre paramètre public local et sa clé secrète principale. Une autorité d'attributs est chargée de vérifier la validité des attributs d'un utilisateur appartenant à son domaine et, si c'est le cas, elle lui émet un composant de la clé secrète en fonction de ses attributs. La clé secrète d'un utilisateur est composée de plusieurs composantes, chacune produite par une autorité d'attribut. De plus, chaque autorité d'attributs se charge de générer périodiquement une clé de mise à jour pour les utilisateurs qui ne sont pas révoqués dans son domaine.
- Propriétaire de données : cette entité représente un propriétaire de données qui souhaite partager ses données en les externalisant vers des serveurs Cloud. Un propriétaire de données définit une politique d'accès basée sur les attributs qu'il met en application sur les données en appelant un schéma CP-ABE multi-autorités. Les données chiffrées ainsi obtenues sont ensuite envoyées aux serveurs Cloud.
- Utilisateur : chaque utilisateur possède un identifiant global unique dans le système, un ensemble d'attributs et une clé secrète regroupant les composants émis par les autorités d'attributs.
- Serveurs Cloud : cette entité est chargée de stocker et de mettre à jour des données chiffrées provenant des propriétaires de données. La mise à jour peut être effectuée en utilisant simplement les paramètres publics du système, y compris le paramètre public global et le paramètre public local de chaque autorité d'attribut, sans implication d'informations secrètes.

Dans cette même thématique, Xiao et al. [Xiao, 15] proposent un système de contrôle d'accès distribué pour sécuriser l'accès aux données hébergées sur le Cloud. Leur système est composé de cinq entités : de multiples autorités d'attributs (AA_1, …, AA_k), les propriétaires de données, les utilisateurs

souhaitant accéder aux données, le serveur Cloud et le serveur d'authentification tiers TP (pour Third Party en anglais). Les utilisateurs souhaitant accéder au système s'authentifient auprès du serveur d'authentification tiers TP. Ce dernier attribue à chaque utilisateur autorisé un identifiant global unique GID. Lors de la révocation d'un utilisateur, son GID est mis à jour par le serveur TP qui maintient la liste des utilisateurs révoqués. Pour chaque utilisateur autorisé ayant un GID actif, le serveur TP distribue une paire de clés globales : la première est secrète $\mathrm{UGSK_{GID}}$ et la deuxième est publique $\mathrm{UGPK_{GID}}$. Le serveur, lui aussi disposant d'une paire de clés secrète $\mathrm{sk_{TP}}$ et publique $\mathrm{pk_{TP}}$, utilise sa clé secrète $\mathrm{sk_{TP}}$ pour générer un certificat d'attribut $\mathrm{ACert_{GID}}$ à partir de la liste d'attributs $\mathrm{AL_{GID}}$ de chaque utilisateur. Pour chaque attribut x d'un utilisateur ayant un GID actif, une autorité d'attributs $\mathrm{AA_k}$ génère une clé publique $\mathrm{APK_x}$ et émet la clé secrète de l'attribut $\mathrm{UASK_{GID,k}}$ à l'utilisateur en fonction de sa liste d'attributs $\mathrm{AL_{GID}}$. Lorsqu'une autorité d'attributs $\mathrm{AA_k}$ reçoit une demande de clé d'un utilisateur, elle vérifie d'abord son certificat d'attribut $\mathrm{ACert_{GID}}$ et ensuite, si la vérification réussit, génère une clé secrète $\mathrm{UASK_{GID,k}}$ et l'envoie au serveur Cloud qui la stocke. Chaque autorité d'attributs $\mathrm{AA_k}$ est autonome et peut gérer n'importe quel nombre d'attributs dans sa capacité. Les propriétaires de données chiffrent leurs propres données avant de les télécharger sur le serveur Cloud. Ce dernier est chargé de stocker les données chiffrées et de fournir un service d'accès aux utilisateurs. Il reçoit et stocke les clés secrètes d'attributs utilisateurs $\mathrm{UASK_{GID,k}}$ des autorités d'attributs. Lorsque le serveur reçoit une demande d'accès à une donnée chiffrée d'un utilisateur avec GID actif, il recherche sa clé secrète d'attribut $\mathrm{UASK_{GID,k}}$, génère et remet à l'utilisateur un jeton de déchiffrement qui lui permet de déchiffrer la donnée.

Par ailleurs, la mise à jour des règles de sécurité dans le chiffrement basé sur les attributs pose de gros problèmes. Par exemple, le changement des attributs par un propriétaire de données implique la mise à jour des règles permettant aux utilisateurs de les déchiffrer. Le plus simple serait que le propriétaire de données les téléchargent du Cloud, les déchiffre, les rechiffre avec les nouveaux attributs et ensuite les republie sur le Cloud. Yang et al. [Yang, 14] proposent un modèle de contrôle d'accès qui supporte la mise à jour dynamique de la politique d'accès, permettant ainsi de contourner ce problème de changement de politique. Ce modèle permet de contrôler l'accès aux données hébergées sur le Cloud tout en supportant la mise à jour dynamique des politiques d'accès. Quatre entités interagissent dans ce modèle :

- Un ensemble d'autorités d'attributs ($\mathrm{AA_1}$, …, $\mathrm{AA_k}$) : indépendantes les unes des autres, chaque autorité d'attributs $\mathrm{AA_i}$ gère les attributs des utilisateurs de son domaine et génère une paire de clés secrète $\mathrm{SK_i}$ et publique $\mathrm{PK_i}$ pour chaque utilisateur en fonction de ses attributs.
- Un serveur Cloud : il stocke les données chiffrées et fournit un service d'accès aux utilisateurs. Il est également responsable de la mise à jour des politiques d'accès aux données chiffrées.
- Les propriétaires de données : ils définissent les politiques d'accès et chiffrent les données sous ces politiques avant de les héberger sur le Cloud. Un message m est chiffré en CT (pour Cipher Text en anglais) en fonction d'un ensemble de clés publiques $\{PK_x\}$ générées par différentes autorités d'attributs et d'une politique de sécurité A : $CT = Enc(m, \{PK_x\}, A)$. La mise à jour des politiques d'accès aux données chiffrées est déléguée au serveur Cloud.
- les utilisateurs souhaitant accéder aux données : un utilisateur peut déchiffrer des données si et seulement si ses attributs satisfont la politique d'accès déterminée par le propriétaire. Un texte chiffré CT peut être déchiffré en texte en clair m par un utilisateur en appliquant un ensemble de clés secrètes $\{SK_x\}$ générées par différentes autorités d'attributs en fonction des attributs de cet utilisateur : $m = Dec(CT, \{SK_x\})$.

Le masquage des politiques d'accès dans le chiffrement basé sur les attributs prend deux formes : il peut être total (toutes les informations de tous les attributs sont masquées) ou partiel (uniquement certaines informations sensibles de certains attributs sont masquées). Plus le nombre d'attributs masqués est grand, plus les données sont mieux protégées mais au détriment de la performance de déchiffrement. Comme expliqué auparavant, dans les schémas CP-ABE, la clé secrète d'un utilisateur est associée à un ensemble d'attributs ; un texte chiffré ne peut être déchiffré que si tous les attributs de l'utilisateur satisfont à la politique d'accès. En pratique, avant de procéder au déchiffrement complet de l'ensemble de données, un "test de déchiffrement" est réalisé en amont pour vérifier si les attributs de l'utilisateur satisfont à la politique d'accès associée au texte chiffré. Dans les schémas CP-ABE masquant la politique d'accès, un utilisateur doit commencer par un test de déchiffrement pour vérifier si ses attributs satisfont ou non à la politique masquée associée au texte chiffré. Ce processus de test de déchiffrement est répété jusqu'à ce qu'une correspondance entre les attributs et la politique d'accès soit trouvée ou que tous les tests possibles aient été effectués. Le coût de calcul dans le test de déchiffrement surcroît linéairement avec la complexité de la politique d'accès et la quantité des informations masquées. Il faut donc trouver le meilleur compromis et ne masquer que les valeurs des attributs les plus sensibles ; les noms des attributs en revanche peuvent généralement ne pas être masqués. Dans ce cadre, Zhang et al. [Zhang, 18-b] présentent le système de contrôle d'accès PASH, appliqué dans le domaine de la santé intelligente (Smart Health), qui prend en considération la protection de la vie privée des individus en masquant les valeurs des attributs à caractères personnels. Le système PASH implique quatre entités :

- Une Autorité d'Attributs : cette entité initialise le système et accorde des privilèges d'accès précis aux utilisateurs en fonction de leurs attributs. L'initialisation du système consiste à générer des paramètres publics du système et une clé principale. Les utilisateurs peuvent ensuite obtenir ces paramètres publics. L'autorisation d'un utilisateur consiste à lui émettre une clé secrète basée sur l'ensemble de ses attributs. De même, un propriétaire de données peut également obtenir sa propre clé secrète si nécessaire.
- Un Serveur Cloud : ce serveur stocke les informations des patients sous un format chiffré et associé à des politiques d'accès partiellement masquées (on ne masque que les valeurs des attributs sensibles).
- Les propriétaires de données : les propriétaires de données gèrent les informations des patients et les transmettent au serveur Cloud sous forme de textes chiffrés. Un propriétaire de données peut être un patient ou un organisme qui gère les informations des patients. Un propriétaire de données peut disposer d'un ensemble d'appareils intelligents qui se composent d'un certain nombre de capteurs portables ou implantables. Les informations collectées de ces capteurs sont cryptées puis envoyées au serveur Cloud. Un propriétaire de données est responsable de la définition et de l'application des politiques d'accès pour les données chiffrées.
- Les utilisateurs : l'utilisateur de données, par exemple un médecin ou un chercheur en médecine, est une personne qui a besoin d'accéder aux informations des patients chiffrées et stockées sur le serveur Cloud. Chaque utilisateur de données possède un ensemble d'attributs et une clé secrète associée à l'ensemble d'attributs. Un utilisateur peut déchiffrer une information si, et seulement si, ses attributs correspondent à la politique d'accès sous-jacente attachée à l'information chiffrée.

3.7.3. Contrôle d'accès pour les environnements intelligents et l'IoT

Un système de transport intelligent ITS (Intelligent Transportation System) est un environnement intelligent qui veille à assurer la sécurité des conducteurs et offrir une expérience confortable aux utilisateurs de l'industrie des véhicules connectés IIoV (Industrial Internet-of-Vehicles). Les infrastructures complexes des ITS ouvrent de larges surfaces d'attaques de sécurité dans le but d'exploiter et de contrôler à distance les mécanismes critiques des véhicules intelligents, y compris les systèmes de moteur et de freinage. Gupta et al. [Gupta, 20] proposent un système formel, baptisé ITS-$ABAC_G$, permettant le contrôle d'accès basé sur les attributs dans l'écosystème IIoV. Le modèle ITS-$ABAC_G$ introduit la notion de groupes pour créer à la volée des ensembles de véhicules connectés en fonction d'un ensemble d'attributs tel que l'emplacement, la direction, la vitesse, etc. ITS-$ABAC_G$ prend en compte les préférences individualisées de chaque utilisateur ainsi que les politiques à l'échelle du système pour accepter ou rejeter les notifications et les alertes des différents véhicules connectés de l'écosystème.

Ouaddah et al. [Ouaddah, 17] proposent le cadre "FairAccess" basé sur la technologie blockchain pour assurer le contrôle d'accès des dispositifs intelligents dans un système IoT. Le contrôle d'accès basé sur FairAccess fonctionne comme suit :

1. Un demandeur *R*, identifié par son adresse *rq*, souhaitant effectuer une action *a* sur une ressource protégée d'un autre dispositif cible, identifié par son adresse *rs*, doit obtenir un jeton d'accès. Pour cela, le demandeur (rq) soumet une requête au propriétaire du dispositif cible *RO* indiquant l'adresse de la ressource cible (rs) et l'action désirée (a).
2. A la réception de la demande, le *RO* définit sa politique de contrôle d'accès : $Policy_{rs,rq}$
3. RO_{Wallet}, le portefeuille du *RO*, agit en tant que point d'application de politique (l'équivalent du PEP du langage XACML) ; il transforme la politique de contrôle d'accès en script de verrouillage (locking script) π_x : $Policy_{rs,rq} \rightarrow \pi_x$ où x est l'empreinte de la transaction *Grant-Access*.
4. Le RO_{Wallet} génère un jeton $TKN_{rs,rq}$ chiffré avec la clé publique du demandeur *R*.
5. Le RO_{Wallet} génère la transaction *Grant-Access* T_x sous la forme suivante : $T_x = (m, sig_{rs}(m))$
   Avec :
   $m = (ID_x, input(rs), output(rq, \pi_x, TKN_{rs,rq}))$; ID_x étant l'index de la transaction T_x.
   $x = H(Tx)$, H est une fonction de hachage.
   La transaction *Grant-Access* est ainsi générée ; elle encapsule la politique de contrôle d'accès définie sous la forme d'un script de verrouillage dans sa sortie, l'adresse *rq* du demandeur et le jeton d'accès signé en tant qu'UTXO (comme expliqué dans la section 3.4.5). Cette transaction est ensuite diffusée par le RO_{Wallet} aux nœuds du réseau, selon le principe pair-à-pair, jusqu'à ce qu'elle atteigne les mineurs qui agissent comme un point de décision de politique (PDP) distribué.
6. Le PDP vérifie la demande avec la politique définie, en comparant le script de déverrouillage (unlocking script) au script de verrouillage (locking script) de la transaction.
7. Si la transaction est valide, le jeton $TKN_{rs,rq}$, représentant sa sortie, sera enregistré dans le blockchain et affiché dans le portefeuille du demandeur (R_{Wallet}). Sinon, la transaction sera rejetée et le demandeur en sera notifié.

Un autre exemple d'utilisation de la blockchain pour contrôler l'accès dans les systèmes IoT est l'architecture ControlChain [Pinno, 17]. Cette architecture est basée sur quatre blockchains :

- **Une blockchain de relations** : cette blockchain est responsable du stockage des informations d'identification publiques et des relations de toutes les entités du système. Dans ControlChain, chaque entité appartient à un propriétaire qui en a le contrôle total dans le système. Une relation d'une entité est une référence unilatérale vers une autre entité avec un ensemble facultatif d'attributs.
- **Une blockchain de contexte** : cette blockchain stocke des informations contextuelles obtenues à partir de capteurs, de données traitées et d'entrées manuelles. Ces informations contextuelles peuvent être utilisées dans la décision d'autorisation.
- **Une blockchain de responsabilité** : cette blockchain stocke des informations sur les autorisations ou les refus d'accès à un objet. Ces informations peuvent être utilisées pour gérer les responsabilités et peuvent être également utilisées comme des informations contextuelles.
- **Une blockchain de règles** : cette blockchain conserve les règles d'autorisation définies par les propriétaires sur leurs objets ou par les objets pour eux-mêmes. Le grand défi de cette blockchain est de la rendre suffisamment générique pour être compatible avec la grande variété des modèles et des mécanismes de contrôle d'accès utilisés dans l'IoT. Les auteurs proposent d'utiliser le design pattern "Décodeur" pour implémenter le processus de transformation entre les modèles de contrôle d'accès. Ce décodeur reçoit le modèle et les règles de contrôle d'accès et les traduit en mécanismes pris en charge dans l'architecture ControlChain.

3.7.4. Contrôle d'accès dans les plateformes Big Data

Apache Hadoop et certains outils de son écosystème tel que Apache Knox, Apache Ranger et Apache Sentry[30] fournissent un système de contrôle d'accès multicouches pour protéger l'accès aux données et aux services d'un cluster Hadoop. Certaines capacités de protection sont fournies par le noyau Hadoop lui-même à travers des listes de contrôle d'accès (ACL) qui spécifient les utilisateurs et/ou les groupes autorisés à effectuer des opérations telles que la soumission d'applications, la requête de l'état des applications, l'accès à des services des démons Hadoop (comme HDFS, NameNode, DataNode et YARN) ou l'accès à des services de l'écosystème Hadoop (comme Apache Hive et Apache HBase). Outre les utilisateurs, les services eux-mêmes ont des accès contrôlés à d'autres services de démons nécessaires pour les mises à jour des tâches ou l'état des ressources du cluster. La Table 3.2 résume un ensemble de listes de contrôle d'accès disponibles dans le package de sécurité du noyau Hadoop (version 3.x).

Table 3.2. Propriétés de configuration d'accès aux Services Hadoop 3.x.

Propriété	Service
security.client.protocol.acl	Cette liste est utilisée pour gérer l'accès des utilisateurs au système de fichiers HDFS.
security.client.datanode.protocol.acl	Cette liste est utilisée pour contrôler l'accès des utilisateurs (les clients) aux blocs des nœuds de données.
security.datanode.protocol.acl	Cette liste est utilisée pour gérer la communication entre les DataNodes et le NameNode.
security.inter.datanode.protocol.acl	Cette liste est utilisée pour la mise à jour de l'horodatage de génération entre les DataNodes.
security.namenode.protocol.acl	Cette liste est utilisée par le NameNode secondaire pour

[30] https://sentry.apache.org/

	communiquer avec le NameNode principal.
security.job.client.protocol.acl	Cette liste est utilisée par les utilisateurs (les clients) pour communiquer avec le gestionnaire de ressources (YARN) pour la soumission des travaux, la demande de statut d'un travail, etc.
security.job.task.protocol.acl	Cette liste est utilisée pour communiquer avec le gestionnaire de nœuds parent pour réduire les tâches.
security.refresh.policy.protocol.acl	Cette liste est utilisée par les commandes dfsadmin et rmadmin pour actualiser la stratégie de sécurité en vigueur.
security.ha.service.protocol.acl	Cette liste est utilisée par HAAdmin (High Availability Admin) pour gérer les états actif et stand-by du NameNode.

En outre, les utilisateurs ne doivent pas être autorisés à consommer toutes les ressources du cluster en exécutant des tâches volumineuses. Apache YARN offre une capacité et des files d'attente équitables qui permettent le partage des ressources entre plusieurs utilisateurs du cluster. Cela empêche tout utilisateur malveillant d'utiliser toutes les ressources du cluster. Une file d'attente est associée à une liste de contrôle d'accès qui détermine les utilisateurs qui sont autorisés à soumettre, à modifier ou à supprimer des applications de la file d'attente. Ces files d'attente sont de nature hiérarchique dans lesquelles les utilisateurs autorisés à soumettre des applications dans les files d'attente parentes, sont également autorisés à soumettre des applications dans les files d'attente enfants, mais pas l'inverse. Hadoop prend également en charge les étiquettes de nœud de cluster au travers desquelles les utilisateurs ne sont autorisés à soumettre des travaux que sur des nœuds de cluster ayant des étiquettes définies en fonction de leurs privilèges.

Par ailleurs, dans l'écosystème Hadoop, de nombreux outils pour renforcer la sécurité des clusters Hadoop ont été développés. Par exemple, Apache Knox, qui est une passerelle applicative, permet d'interagir à travers un point d'accès unique avec les APIs REST et les interfaces des utilisateurs d'un ou plusieurs clusters Hadoop. Cette couche, en tant que telle, est la première couche de contrôle d'accès ; elle permet de vérifier et d'empêcher les utilisateurs d'accéder aux services bien avant l'accès aux données. Une fois que l'utilisateur est autorisé à accéder aux services, le contrôle d'accès aux données est pris en compte. HDFS est chargé de stocker des fichiers volumineux de manière distribuée sur différents nœuds de données ; il utilise principalement des listes de contrôle d'accès (certaines sont listées dans la Table 3.2) sur les fichiers et les répertoires pour se protéger des accès non-autorisés. Hadoop permet l'accès aux mêmes données résidant dans HDFS dans différents formats en fonction du modèle de données pris en charge. Par exemple, Hive représente les données dans des tables et des colonnes, HBase prend en charge les familles de colonnes, etc. Ces modèles de données prennent en charge différentes opérations telles que la sélection et la suppression dans Hive, la lecture et l'écriture dans HBase, etc. Apache Ranger et Apache Sentry fournissent des plugins qui sont attachés à ces services pour appliquer des contrôles d'accès basés sur des règles. Par conséquent, lorsque les données sont accessibles dans un service comme Apache Hive, l'autorisation est vérifiée à la fois sur les objets Hive et HDFS, en plus de la vérification préalable du périmètre d'accès au service. Apache Ranger introduit également la notion de balises d'objets, où les objets de données se voient attribuer certaines valeurs d'attribut, et le contrôle d'accès est défini en fonction de ces balises. Apache Ranger permet également aux stratégies d'activer le masquage de colonnes et le filtrage de lignes dans Apache Hive où certains ensembles de données sont masqués ou filtrés des utilisateurs qui ne sont pas autorisés à y accéder. Apache Sentry, quant à lui, permet d'appliquer des autorisations à base de rôles avec une granularité fine aux données stockées sur un cluster Hadoop.

Gupta et al. [Gupta, 17] proposent le modèle HeAC (Hadoop ecosystem Access Control) qui est une formalisation du système de contrôle d'accès implémenté dans le noyau Hadoop ainsi qu'Apache Ranger et Apache Sentry. OT-RBAC [Gupta, 17] (Object-Tagged RBAC) et HeABAC [Gupta, 18] (Hadoop ecosystem ABAC) sont deux extensions du modèle HeAC qui lui intègrent la gestion des autorisations à base des rôles (dans OT-RBAC) et à base d'attributs (dans HeABAC). Vu que le principe est le même (introduction des rôles dans OT-RBAC et des attributs dans HeABAC), nous nous contentons dans cette section de présenter le modèle OT-RBAC dont les principaux composants sont :

- **Utilisateurs (U), Groupes (G), Rôles (R) et Sujets (S)** : les auteurs définissent un utilisateur comme étant un humain qui interagit directement avec le système informatique pour accéder aux objets de données et aux services dans un cluster Hadoop. Les utilisateurs peuvent avoir des caractéristiques et des exigences similaires qui sont regroupées pour former des groupes. Un rôle est une collection d'autorisations attribuées à divers utilisateurs dans le système. Un sujet représente une application exécutée au nom de l'utilisateur pour effectuer des opérations réelles sur les objets et les services dans l'écosystème Hadoop.
- **Services Hadoop (HS)** : il s'agit des services représentés par des démons (i.e. des processus) qui s'exécutent en arrière-plan pour fournir les fonctionnalités de base de la plateforme Hadoop. Parmi ces services on trouve HDFS, NameNode, DataNode, YARN, et bien d'autres. Les utilisateurs et les sujets interagissent avec les démons pour soumettre une application ou pour afficher l'état des travaux en cours. Ces services nécessitent une interaction entre eux pour les opérations, y compris les mises à jour des tâches ou la surveillance des ressources.
- **Opérations sur les services Hadoop (OP_{HS})** : il s'agit de l'ensemble d'actions autorisées sur les démons Hadoop, comprenant principalement les opérations d'accès effectuées par les utilisateurs ou d'autres services (par exemple, le NameNode récupère des informations des DataNodes).
- **Services de l'écosystème (ES)** : l'écosystème Hadoop prend en charge plusieurs projets ou services tels qu'Apache Hive, Apache HBase, etc. qui ont des données sous-jacentes ou des objets de services exploités par les utilisateurs. Tout utilisateur ou sujet doit être autorisé à accéder au service avant de pouvoir accéder à ses objets. Par exemple, l'utilisateur doit d'abord autoriser l'accès au service Apache Hive avant que toute opération sur une table Hive ne soit effectuée.
- **Objets de données et de services (OB)** : différents services de l'écosystème Hadoop prennent en charge divers objets de données et de services qui sont exploités par les utilisateurs ou les sujets. Ce sont les ressources réelles qui sont protégées contre les accès non-autorisés dans le cluster Hadoop (par exemples, les objets en file d'attente dans YARN, les familles de colonnes dans Apache HBase, les tables dans Apache Hive, etc.).
- **Opérations sur les objets (OP)** : différents objets dans les services de l'écosystème Hadoop prennent en charge diverses opérations au sein du cluster Hadoop en fonction du modèle de données pris en charge. Les opérations peuvent être des actions de lecture et d'écriture dans HDFS, de soumission et d'administration des objets dans des files d'attente YARN, etc.
- **Balises d'objets (Tag)** : il s'agit des valeurs d'attributs attachées à divers objets du système. Ces valeurs peuvent aider à définir des politiques basées sur des balises (ou tags) dans lesquelles un utilisateur est "autorisé" ou "refusé" d'effectuer une opération sur une ressource en fonction de la balise associée à la ressource.

Deux types d'autorisations existent dans le modèle OT-RBAC : les autorisations de services Hadoop (**HS-PRMS**) et les autorisations d'objets (**OBJECTS-PRMS**). Les HS-PRMS définissent les

opérations sur les services Hadoop et sont formellement déclarées comme un ensemble de puissance du produit croisé de HS et OP_{HS}. Les OBJECT-PRMS, quant à elles, définissent l'ensemble des permissions pour effectuer des opérations (OP) sur les objets (OB) dans les services de l'écosystème (ES). Ces autorisations incluent soient des objets, soient des balises associées aux objets pour créer des politiques basées sur des balises. L'accès à un service de l'écosystème est d'abord requis avant d'effectuer une opération sur ses objets. Pour effectuer des opérations dans un cluster, il peut être nécessaire pour un utilisateur de disposer d'une ou des deux autorisations.

Un autre travail étendant le système Hadoop est celui de Chen et al. [Chen, 14-b] dans lequel ils proposent un modèle de contrôle d'accès basé sur des "labels multiples" pour protéger les données stockées dans HDFS. Pour stocker un fichier dans HDFS, il est tout d'abord découpé en un ensemble de blocs qui sont ensuite stockés dans les DataNodes. Le NameNode détient les métadonnées qui permettent d'identifier les blocs et leurs réplications. Les auteurs proposent d'étendre la structure des métadonnées pour enregistrer des labels. Ils proposent d'utiliser la solution QueryIO[31] pour étendre les métadonnées du NameNode pour en ajouter d'autres comme :

- Niveau de sécurité : inspiré du modèle de contrôle d'accès obligatoire "Sécurité multi-niveau", les auteurs proposent d'intégrer différents niveaux de sécurité selon les types de données. Par exemple, dans le cadre d'un projet de santé, un fichier qui inclut le numéro de sécurité sociale d'un patient doit avoir le niveau de sécurité "Secret".
- Durée de vie : cette notion représente le temps d'efficacité d'une information et par conséquent le temps d'existence de la donnée dans HDFS. A l'expiration de ce temps, la donnée ne doit plus exister dans HDFS ; elle doit être supprimée. Par exemple, on peut déterminer la durée de vie d'un rapport d'examen médical à 36 mois.
- Valeur de hachage : les auteurs de l'article utilisent une valeur de hachage pour protéger l'intégrité des labels. Pour cela, ils utilisent l'algorithme de hachage SHA1.

Avant d'écrire des données, leurs propriétaires doivent s'authentifier par le protocole Kerberos et ensuite renseigner les différents labels. Une fois que ces labels sont définis, le système de contrôle d'accès les protégera. Chaque entité souhaitant accéder, en lecture ou en écriture, à une donnée dans HDFS doit s'authentifier par le protocole Kerberos pour obtenir un niveau de privilège de sécurité en fonction de ses privilèges (rôle ou habilitation) dans le système. Si les attributs de l'entité et les labels répondent aux règles de contrôle d'accès, l'entité peut accéder aux données, sinon l'accès est refusé. Le modèle proposé dans [Chen, 14-b] consiste à étendre le système Hadoop tout en permettant aux propriétaires de données, lors de la création des fichiers, de déclarer des labels comme étant des nouvelles métadonnées du NameNode. Ces labels vont pouvoir ensuite contrôler les entités souhaitant accéder aux données en croisant les niveaux de privilèges des entités avec la politique de contrôle d'accès aux données.

D'autres travaux s'intéressent à l'intégration de la gestion de la confiance dans le système Hadoop. On y trouve notamment le schéma TDACS (Trust-based Dynamic Access Control Scheme) [Yang, 20] qui gère l'accès à la plateforme Hadoop non seulement à travers les droits d'un utilisateur mais aussi en fonction de la confiance du système en cet utilisateur. La confiance peut être calculée en fonction de l'historique du comportement de l'utilisateur. Pour cela, TDACS intègre un proxy d'accès qui interagit avec deux interfaces :

[31] http://queryio.com/

- La première interface, appelée "autorisation basée sur ABAC", implémente un système de contrôle d'accès basé sur les attributs pour gérer l'autorisation frontale d'accès à la plateforme.
- La deuxième interface, appelée "autorisation basée sur la confiance", assure la granularité fine de l'autorisation en maintenant une liste composée de la valeur de seuil de confiance fournie pour chaque ressource.

3.8. Discussion

Probablement, l'accès à l'information confidentielle représente le défi le plus complexe et le plus préoccupant en Big Data. Outre les bonnes pratiques à tenir en considération pour éviter des attaques de type Phishing, Spamming ou Spoofing, de nombreux mécanismes doivent être mis en œuvre pour protéger les systèmes d'information. Parmi ces mécanismes, les plus discutés dans la littérature sont l'authentification des utilisateurs, le chiffrement des données sensibles et le contrôle d'accès aux données, aux ressources et aux services. L'authentification permet de créer un environnement de confiance dans lequel chaque utilisateur dispose d'un identifiant unique lui permettant de s'authentifier pour accéder au système. En cas d'authentification par mot de passe, une politique de gestion des mots de passe doit être appliquée ; certains exemples de politiques sont fournis par la CNIL [CNIL, 18-b]. Dans les systèmes distribués, comme le recommande CSA [CSA, 16], l'authentification Kerberos, ou équivalent, est nécessaire. Concernant le chiffrement, même s'il présente l'avantage de protéger les données en cas d'accès non-autorisé, les cadres distribués comme Hadoop et les bases de données NoSQL tentent souvent de stocker les données dans leurs états bruts, sans les chiffrer, dans le but de gagner en efficacité de traitement. Certains travaux de recherche comme [Inukollu, 14] recommandent de crypter toutes les données stockées sur les nœuds d'un cluster Hadoop en protégeant les clés de chiffrement derrière des pare-feux puissants et dans des machines différentes. [Park, 13] proposent d'étendre le cadre Hadoop en y intégrant une fonction de cryptage et décryptage lors de l'écriture de données dans HDFS et la lecture de données par MapReduce. Le cryptage de l'ensemble de données stockées sur HDFS est techniquement possible mais risque de poser de nombreux soucis à la robustesse, la performance et la fiabilité des calculs. A titre d'exemple, la révocation d'un seul utilisateur est une tâche très complexe et nécessite l'intervention du propriétaire de données pour recrypter toutes les données, générer de nouvelles clés secrètes et les partager avec les autres utilisateurs toujours actifs. Aussi, l'exécution des requêtes et des calculs sur des données cryptées ne semble pas être efficace (le débogage et les plans d'exécution des requêtes ne peuvent pas aider à interpréter les résultats). Enfin, concernant le contrôle d'accès, toutes les études récentes relèvent le besoin d'automatisation de la prise de décision et de la granularité fine des systèmes de contrôle d'accès. Les règles de sécurité portées sur des tables d'une base de données ou des répertoires entiers ne permettent pas de garantir une confidentialité suffisante de données. En effet, les utilisateurs ayant accès à une table d'une base de données peuvent abuser de ce privilège pour accéder à des enregistrements ne les concernant pas. Les règles d'accès doivent en conséquence être décidées de façon automatique et en fonction des utilisateurs et leurs profils, d'une part, et du contexte d'utilisation et le contenu de données, d'autre part. De nombreuses extensions des modèles de contrôle d'accès classiques comme RBAC et ABAC ont été proposées dans la littérature et semblent apporter des solutions à cette problématique. Toutefois, à ce jour, aucun des modèles n'est standardisé ou reconnu par une grande communauté ; la majorité de ces modèles souffrent de nombreuses limites dues principalement à la grande volumétrie, à l'hétérogénéité et à la vitesse de génération et de traitement des données en Big Data.

Par ailleurs, les systèmes de gestion des bases de données relationnelles fournissent des fonctionnalités de sécurité étendues telles que le chiffrement de données, la gestion des configurations des utilisateurs, l'authentification et le contrôle d'accès, tandis que les bases de données non-relationnelles sont privées des fonctionnalités de sécurité de bases telles que GRANT et REVOKE. Initialement, les solutions Big Data, comme Hadoop et son écosystème, ont été conçues pour construire des canalisations (ou pipelines) de traitement de données performantes plutôt que sécurisées [RezaeiJam, 14], [Dincer, 17]. A ce jour, de nombreux risques de sécurité sont encore associés aux plateformes Hadoop ; à titre d'exemple, lors du transfert de données sur le réseau d'un client Hadoop vers les DataNodes, le risque de fuite de données se présente. En plus, comme les nœuds du système de stockage et de traitement échangent souvent des données, le risque de violation de la confidentialité de données se pose. Cela nécessite des mécanismes de sécurité solides. Par ailleurs, avec l'émergence du Cloud Computing, Hadoop a évolué vers la Big Data as-a-Service ce qui signifie que les données sensibles des entreprises sont maintenant stockées sur le Cloud public et tous les services sont accessibles via Internet. Cela oblige les organisations à faire face à de nombreux problèmes liés aux fuites de données et à la sécurité en général. Les fournisseurs du Cloud proposent des infrastructures qui répondent aux exigences fonctionnelles de leurs clients. Pour que ces infrastructures soient rentables et évolutives, les fournisseurs doivent partager des périphériques de stockage et des ressources physiques entre plusieurs clients ; c'est ce qu'on appelle la multi-location (ou multi-tenant). Toutefois, d'un point de vue sécurité, le partage des ressources signifie que les ressources sont sujettes aux attaques. Si le client et l'attaquant utilisent les mêmes appareils physiques, et si les mesures de sécurité appropriées ne sont pas mises en œuvre, l'attaquant pourrait accéder aux données des clients ou juste perturber leur disponibilité en monopolisant les ressources partagées. Ce problème devient encore plus critique étant donné que les entreprises n'ont pas de contrôle direct sur leurs données [Parmar, 17] ; elles ne peuvent jamais savoir si leurs données sont utilisées par quelqu'un d'autre ou non. Comme les clients doivent partager des ressources physiques avec d'autres clients et qu'ils n'ont pas de contrôle direct sur leurs données, ils n'ont plus de choix que de s'appuyer sur la confiance qu'ils ont en leurs fournisseurs de Cloud [Kapil, 18].

Pour combler ces limites, de nombreux groupes de recherche travaillent sur le renforcement de la sécurité de données sur le Cloud Computing, la plateforme Hadoop et les infrastructures Big Data en général. Dans le domaine du Cloud Computing, Xu et al. [Xu, 12] proposent un schéma de rechiffrement de proxy, sans certificat, pour le partage sécurisé de données avec le Cloud public. Appelé CL-PRE (Certificate-Less Proxy Re-Encryption), ce schéma intègre de manière unique la clé publique basée sur l'identité dans le rechiffrement proxy. Dans leur approche, les propriétaires de données partagent des données avec des utilisateurs du Cloud (les destinataires). Une donnée est d'abord chiffrée avec une clé de chiffrement symétrique DEK (Data Encryption Key) par son propriétaire, puis stockée dans le Cloud avec une liste de contrôle d'accès ACL (Access Control List) indiquant le groupe de destinataires. Le propriétaire de données chiffre également la DEK, à l'aide de sa clé publique, et l'envoie sur le Cloud. À la demande d'accès d'un destinataire, basée sur l'ACL, un serveur proxy dans le Cloud prend une clé de rechiffrement envoyée par le propriétaire de données et utilise un algorithme de rechiffrement pour transférer la DEK chiffrée dans le format qui peut être déchiffré par la clé privée du destinataire. Le destinataire peut alors télécharger les données chiffrées et utiliser la DEK pour le déchiffrement. Le schéma CL-PRE élimine le problème de séquestre de clés dans le chiffrement traditionnel basé sur l'identité et ne nécessite pas l'utilisation de certificats pour garantir l'authenticité des clés publiques.

Park et Lee [Park, 13] ont proposé une architecture Hadoop sécurisée en ajoutant des fonctions de chiffrement et de déchiffrement dans HDFS. L'écriture d'un fichier sur HDFS inclut une étape de chiffrement dans le processus d'écriture par défaut de Hadoop. Un client HDFS divise un fichier en un ensemble de blocs, chiffre chaque bloc et les sauvegarde ensuite dans HDFS. La fonction de chiffrement elle-même peut être implémentée en surchargeant une classe Java, appelée AESCodec. Le nouveau module de chiffrement AES-Encryption, basé sur la classe AESCodec, exécute l'algorithme de chiffrement sur le CPU. Le client HDFS exécute la classe AESCodec pour effectuer le chiffrement et pour transmettre les blocs HDFS chiffrés à un nœud de données. Le premier nœud de données qui reçoit les blocs HDFS chiffrés les diffusera ensuite vers les autres nœuds de données pour la réplication.

G-Hadoop [Wang, 13] est une extension du cadre Hadoop permettant aux tâches MapReduce de s'exécuter sur plusieurs clusters. L'architecture de haut niveau du cadre G-Hadoop suit le modèle de communication maître/esclave mais à une échelle plus grande que celle de Hadoop. En effet, les DataNodes dans le modèle Hadoop sont remplacés par des clusters dans le modèle G-Hadoop. Le nœud maître est l'entité centrale de l'architecture G-Hadoop. Il est chargé d'accepter les travaux soumis par l'utilisateur, de diviser les travaux en tâches plus petites et de répartir ces tâches entre ses nœuds esclaves. Il est également responsable de la gestion des métadonnées de tous les fichiers disponibles dans le système. Le nœud maître contient un serveur de métadonnées qui gère les fichiers distribués entre plusieurs clusters et un JobTracker qui est responsable de la division des travaux en tâches plus petites et de la planification de ces tâches entre les clusters participants pour exécution. Les tâches sont soumises à la file d'attente du planificateur de cluster (TORQUE) à l'aide d'une interface standard. Le nœud esclave de G-Hadoop permet l'exécution des tâches MapReduce sur les nœuds de calcul des clusters participants. Son composant principal est un TaskTracker qui est chargé d'accepter et d'exécuter des tâches à partir du JobTracker du nœud maître. Toutefois, G-Hadoop réutilise le mécanisme d'authentification des utilisateurs et de soumission de tâches de Hadoop qui est conçu pour un seul cluster. Le mécanisme de sécurité dans G-Hadoop applique le protocole Secure Shell (SSH) pour établir une connexion sécurisée entre un utilisateur et le cluster cible. Cette approche nécessite une connexion unique à chaque cluster participant et les utilisateurs doivent se connecter à chaque cluster pour être authentifiés. Pour combler cette limite, Zhao et al. [Zhao, 14] proposent un nouveau modèle de sécurité pour G-Hadoop. Adapté pour les environnements distribués, ce modèle de sécurité est basé sur un ensemble de mécanismes tels que le chiffrement à clé publique et le protocole SSL. Le composant supplémentaire de cette architecture est le serveur CA (Certificate Authority), qui est nécessaire pour émettre des informations d'identification de proxy ainsi que des informations d'identification des esclaves. Il s'agit donc d'une extension du cadre G-Hadoop impliquant des étapes supplémentaires pour authentifier les parties de communication et établir une connexion sécurisée avant d'exécuter des tâches.

De nombreuses solutions ont été donc développées au fil des années pour renforcer la sécurité de données dans les plateformes Big Data. Il est désormais possible d'intégrer de nombreux mécanismes de sécurité dans un cluster Hadoop. Tout d'abord, les distributions Hadoop ont effectué une grande partie du travail d'intégration et de configuration avec une sécurité centrale comme Active Directory ou LDAP via Apache Knox[32]. Il s'agit d'un système qui fournit un point unique d'authentification et d'accès pour le service Hadoop. Il permet l'accès via les protocoles HTTP et HTTPS au cluster Hadoop et élimine ainsi

[32] https://knox.apache.org/

les risques de nœuds de périphérie[33] SSH. Pour sécuriser la communication entre les différents nœuds d'un cluster Hadoop, de nombreux protocoles d'authentification peuvent être appliqués comme Kerberos. Des techniques de hachage d'authentification ont été également mises en œuvre [RezaeiJam, 14]. Par ailleurs, le chiffrement HDFS a été intégré dans le système de fichiers Hadoop ce qui signifie que les données peuvent être chiffrées lors de leur stockage dans le système de fichiers, de manière transparente, sans modification des applications qui utilisent le cluster. Il s'agit d'une fonctionnalité importante pour assurer la confidentialité des données des colocataires dans les clusters multi-locataires. HDFS peut également être utilisé avec le service de gestion de clés KMS (Key Management Service) de Hadoop ou intégré à des services de gestion de clés tiers [Kapil, 18].

Nous révélons également que certains mécanismes de sécurité risquent d'être mutuellement bloquants. Un premier exemple concerne les mécanismes de chiffrement, pour assurer la confidentialité de données, et ceux de déduplication, pour assurer l'intégrité de données. L'objectif du chiffrement est de protéger les données contre les accès non-autorisés. Si les données sont chiffrées correctement, même en cas d'accès non-autorisé, elles restent protégées puisqu'elles sont illisibles et difficilement exploitables. Cependant, comme le confirme [Li, 14-a], les techniques de chiffrement traditionnelles sont incompatibles avec la déduplication de données. En effet, le chiffrement traditionnel consiste à ce que des utilisateurs différents chiffrent leurs données avec leurs propres clés. Ainsi, des copies de données identiques de différents utilisateurs conduiront à des textes chiffrés différents, rendant ainsi la déduplication impossible. Cela peut avoir un intérêt pour préserver la confidentialité de données appartenant à chaque utilisateur mais ne résout pas la question de déduplication de données. En outre, le chiffrement de l'ensemble de données peut avoir un impact sur l'intégrité des données et l'efficacité des traitements. En effet, un traitement effectué sur des données chiffrées pose des défis de faisabilité, de fiabilité des résultats et de rapidité des traitements ; travailler sur des données claires est nettement plus simple, plus rapide, plus efficace et plus sûr que travailler sur des données chiffrées. Enfin, dans le contexte Big Data, chiffrer de grandes masses de données ne semble pas être une bonne solution vis-à-vis de la disponibilité des données et du système. Chiffrer de grandes masses de données nécessite la gestion d'un très grand nombre de clés de chiffrement, sollicite les processeurs pour de longues opérations de chiffrement et de déchiffrement et rend la gestion des autorisations (GRANT et REVOKE) encore plus complexe. En conclusion sur ce point, chiffrer l'ensemble de données est un processus qui peut impacter la déduplication des données, l'efficacité des traitements et la disponibilité des données et des services. Ainsi, il est primordial d'identifier les données méritant d'être chiffrées avec des mécanismes distribués pour la gestion des clés.

Un deuxième exemple de blocage mutuel entre les mécanismes de sécurité concerne la déduplication et la disponibilité de données. L'objectif de la déduplication, outre la libération de l'espace de stockage et la bande passant du réseau, est de supprimer les doublons pour améliorer la cohérence des données et préserver leur intégrité. En effet, l'existence de plusieurs copies d'une même donnée nécessite la mise en place de certains mécanismes de synchronisation lors de toute modification de cette donnée. Ces mécanismes de synchronisation peuvent impacter la latence du système puisqu'ils sollicitent les processeurs d'effectuer des opérations supplémentaires. En revanche, en Big Data, comme les données et les traitements sont distribués sur différents nœuds, le risque de perdre des données est trop élevé. Si les données ne sont ni répliquées, ni archivées ou si elles ont été mal dédupliquées, leur récupération en cas

[33] Un nœud de périphérie (ou nœud de passerelle) fonctionne comme un portail côté utilisateur final pour la communication avec d'autres nœuds dans un cluster.

de perte devient impossible. En outre, dans les plateformes Big Data comme Hadoop, HDFS fournit un mécanisme de réplication des blocs de fichiers qu'il gère. Pendant la phase d'écriture, chaque bloc est répliqué sur plusieurs nœuds de données. Pour la phase de lecture, si un bloc est indisponible sur un nœud de données, le nœud principal (NameNode) cherche une copie de ce bloc sur les autres nœuds de données. En conclusion sur ce deuxième point, la déduplication est un processus requis pour améliorer la cohérence et l'intégrité de données mais peut impacter la disponibilité de données en cas de perte. Un compris doit être mis en œuvre pour régler tous ces conflits.

Un troisième exemple concerne la protection de la vie privée. Bien que les gouvernements se penchent de plus en plus sur ce sujet, le respect des contraintes légales présente de nombreux problèmes. Dans certains cas, les réglementations peuvent elles-mêmes entraîner une violation de la vie privée. Par exemple, certaines réglementations permettent de conserver les données pendant un certain temps, quelques années en général, même après la fin du projet pour lequel ces données ont été collectées. D'un point de vue sécurité, tant que l'on conserve ces données personnelles, il existe un risque de violation de la vie privée. Un excès de réglementations peut également empêcher la bonne exploitation des données.

Au fur et à mesure que les technologies Big Data mûrissent, la vaste collecte des données personnelles soulève des préoccupations sérieuses pour les particuliers, les entreprises et les gouvernements. L'utilisation et le partage des données privées des utilisateurs manquent de spécifications ; cela réduit leur confiance envers les entreprises qui exploitent ces données. Sans répondre à ces préoccupations, les individus peuvent trouver l'analyse de données inquiétante et décider de ne pas fournir des données personnelles afin d'être analysées par des tiers. Cela peut être contre-productive tant pour les entreprises que pour les clients, car les technologies Big Data sont aujourd'hui essentielles pour réduire les coûts et améliorer la qualité de vie. Par conséquent, les entreprises et les clients doivent trouver un équilibre entre l'utilisation des données personnelles pour les services et les problèmes de protection de la vie privée. Nous pouvions constater cette problématique pendant la pandémie de COVID-19. Dans certains pays, la recherche des contacts avec des personnes identifiées comme porteuses du coronavirus était une procédure récursive basée sur la mémoire des personnes testées positives. Dans d'autres pays, grâce aux technologies Big Data, des applications de traçage des contacts (Contact-Tracing Apps) ont été développées. Bien que ces applications répondent aux exigences des réglementations en matière de protection de la vie privée, le nombre de personnes qui ont accepté de les utiliser était trop faible. Cela ne peut s'expliquer que par le manque de confiance des individus en ces applications. Dans [Talha, 20-c], nous avons proposé une approche qui donne le contrôle des informations personnelles aux personnes concernées plutôt qu'aux organismes gérant ce type d'applications. Il est à noter, pour terminer ce point, qu'il n'y a pas de mesure unique pour la protection de la vie privée ; l'équilibre dépend du type de service, des personnes servis, du type de données et des environnements réglementaires [Lee, 17].

Malgré toutes ces avancées, les infrastructures et les solutions Big Data souffrent encore de nombreux problèmes et limites de sécurité que ce soit au niveau technique (mécanismes et protocoles), conceptuel (architectures et modèles) ou réglementaire (normes et lois). Le plus grand défis des plateformes Big Data réside toujours dans l'implémentation d'un système de sécurité robuste, fiable et respectant la réglementation en vigueur, sans pour autant impacter les performances de stockage, de traitement et de calcul.

3.9. Conclusion

La Big Data est en train de devenir un domaine émergent pour tous les secteurs économiques et sociaux. Elle améliore non seulement l'avantage concurrentiel des entreprises mais aussi la qualité de vie des individus. Grâce à cette émergence technologique, le monde entier "se digitalise" ; il passe de la fabrication humaine à l'automatisation. Ce changement technologique n'est toutefois pas sans risque. La sécurité de données est l'une des préoccupations majeures des projets Big Data. Le contexte volumineux, distribué, extensible et hétérogène des environnements pose de nombreux défis et problèmes à la sécurité de données. Ce chapitre présente un état de l'art sur les grands défis et risques de sécurité ainsi que les recommandations à tenir en considération pour limiter ces risques. Il traite la sécurité de données dans Hadoop qui est aujourd'hui en passe de devenir le standard de facto de stockage et de traitement des données volumineuses. Pendant plusieurs années, cette plateforme a souffert d'un faible système de sécurité. Aujourd'hui, le noyau d'Apache Hadoop et certains outils de son écosystème tels qu'Apache Ranger, Apache Sentry et Apache Knox ont nettement amélioré le système de sécurité, mais, malgré cela, ils restent insuffisants pour couvrir tous les besoins de sécurité comme la fusion des politiques de sécurité et le contrôle d'accès à granularité fine. Certains travaux de recherche, présentés dans ce chapitre, tentent de renforcer le système de sécurité de Hadoop mais, à notre connaissance, aucun n'a été adopté par une large communauté ou reconnu comme une solution efficace.

Par ailleurs, le contrôle d'accès est considéré aujourd'hui comme un mécanisme indispensable pour que les organisations protègent efficacement les données et les ressources de leurs systèmes d'information. La recherche sur les modèles de contrôle d'accès a réalisé des progrès notables après de longues années de développement. D'autres domaines centrés sur la donnée tels que le Cloud Computing et l'IoT ont également participé à l'évolution des modèles de contrôle d'accès et ont rendu ce sujet l'un des problèmes les plus importants de la recherche actuelle sur la sécurité de données. De nombreux modèles proposés récemment dans la littérature abordent certaines problématiques telles que la distribution, la virtualisation, la multi-location, l'automatisation des accès, etc. Une grande attention a été attirée à ce sujet par les universitaires et les industriels. Par conséquent, il est attendu que les études futures mettent davantage l'accent sur la conception des modèles de contrôle d'accès dans les environnements Big Data.

Chapitre 4. Evaluation de l'exactitude de données en Big Data

4.1. Introduction

La Qualité, dans un cadre général, est définie comme étant la conformité aux exigences [Crosby, 79]. Pour les données, la qualité représente le degré auquel un ensemble de caractéristiques de données remplit les exigences [ISO-IEC 25012, 06]. C'est un sujet essentiel pour toute organisation pour obtenir des données exactes et d'en prendre en conséquence de bonnes décisions. Les préoccupations concernant ce sujet peuvent être abordées de trois manières : l'évaluation de données, l'amélioration de données et la protection de données [Motro, 98]. Nous dédions ce chapitre à l'évaluation de l'exactitude de données (Data Accuracy, en anglais) dans le contexte Big Data. Il s'agit d'une des principales dimensions de la qualité de données. Elle mesure le degré auquel les données sont correctes. Connaitre le degré d'exactitude de données, que ce soit élevé ou bas, reflète le niveau de confiance que l'on peut leur attribuer lors des processus de prise de décision. Toutefois, l'évaluation de l'exactitude de données n'est pas une tâche simple à réaliser. La complexité réside dans l'obtention des données correctes, servant de données de référence, avec lesquelles il faudra comparer les données que l'on souhaite évaluer.

Les dimensions les plus étudiées dans la littérature sont l'Exactitude, l'Exhaustivité, la Cohérence, l'Actualité et la Pertinence [Redman, 96] [Fugini, 02]. Plusieurs études ont identifié l'exactitude comme étant la dimension clé de la qualité de données [Wand, 96] [Motro, 98] [Batini, 06] [Redman, 05]. Connaitre au préalable l'exactitude de leurs données apporte de nombreux avantages aux organisations. Les données inexactes peuvent conduire à des conclusions erronées et peuvent compromettre considérablement toute une gamme de processus de prise de décision ce qui peut entrainer des pertes d'opportunités, des pertes de revenus, des erreurs stratégiques, etc. Ainsi, l'évaluation de l'exactitude de données est un processus qui nécessite une planification avant toute exploitation de données.

De nombreux travaux dans la littérature tentent de trouver des solutions pour évaluer l'exactitude de données. La majorité de ces travaux requiert une étape principale qui consiste à comparer les données à évaluer avec des données correctes sans pour autant détailler suffisamment comment obtenir ces données de référence. Aujourd'hui, grâce à l'émergence de la Big Data, le Cloud Computing et l'IoT, les organisations peuvent facilement collecter, stocker et gérer de très gros volumes de données. Dans ce chapitre, nous proposons une solution pour récupérer des données de référence, qui serviront ensuite comme sources de comparaison des données dont on souhaite évaluer l'exactitude.

Après avoir présenté une vue d'ensemble sur les travaux qui s'intéressent à l'évaluation de l'exactitude de données, nous allons détailler la notion de « l'exactitude de données » afin de cadrer une définition qui nous servira de base pour le reste de ce travail. Nous présenterons ensuite un ensemble de techniques et de concepts nécessaires à la mise en œuvre d'un processus d'évaluation de l'exactitude de données. Nous détaillerons par la suite notre modèle et architecture d'évaluation de l'exactitude de données en Big Data. L'analyse des résultats de l'étude de cas nous permettra de déduire un ensemble de constats très intéressants pour réussir l'implémentation de notre solution pour tout autre projet.

4.2. Travaux similaires

De nombreux travaux s'intéressent à l'évaluation de la qualité de données notamment à travers l'étude de leur degré d'exactitude. Dans [Motro, 98], Motro et Rakov présentent une solution d'évaluation

de l'exactitude et de l'exhaustivité des bases de données relationnelles. Ils considèrent qu'une base de données D et que le monde réel qu'elle modélise W peuvent être formalisés comme deux instances d'un même schéma de base de données et que l'instance D est une approximation de l'instance W. Pour déterminer la validité de cette approximation, il faut mesurer la similarité de ces deux instances. Comme chaque instance est un ensemble de tuples, ils proposent d'utiliser des mesures qui comparent deux ensembles d'éléments à travers la formule suivante : $\frac{|D \cap W|}{|D|}$ avec $|X|$ dénote la cardinalité d'un ensemble X. Cette formule semble être limitée en termes de précision de calcul. Une première amélioration présentée par les auteurs eux-mêmes consiste à décomposer toutes les relations en unités d'information plus petites. Une relation de n attributs $A_1, \ldots, A_n$ dans laquelle A_1 est l'attribut clé, est décomposée en n instances de deux attributs (A_1, A_i), avec i allant de 1 à n. La comparaison d'instances décomposées fournit des mesures de qualité plus affinées. Comme les sources d'informations sont rarement de qualité uniforme, certaines peuvent fournir des informations d'excellente qualité sur des sujets mais pas sur d'autres. Ainsi, ces deux méthodes d'évaluation risquent de fournir des estimations grossières de l'exactitude de données.

Redman dans [Redman, 05] propose un cadre permettant de choisir les mesures de l'exactitude de données selon quatre facteurs. Le premier facteur concernant « l'endroit où les mesures sont prises » consiste à déterminer l'endroit le plus approprié pour appliquer les mesures de qualité (chez le fournisseur de données lors de la transmission de celles-ci, lors de l'insertion des nouvelles données dans une base de données, dans la base de données elle-même, lors de la livraison aux consommateurs de données, lors de la réception par le consommateur ou sur toute la chaine de l'information). Le deuxième facteur concernant « les données à inclure dans les mesures» consiste à déterminer si la mesure de l'exactitude doit se contenter des attributs clés, c'est-à-dire les champs les plus importants pour une utilisation particulière ou inclure l'ensemble de données, notamment lorsqu'il est difficile de déterminer lesquelles des données sont les plus importantes. Le troisième facteur concernant « le dispositif de mesure » consiste à déterminer comment mesurer l'exactitude de données (en faisant appel aux experts, en comparant leurs valeurs avec celles du monde réel, en comparant leurs valeurs avec leurs domaines de valeurs autorisés, en traitant les plaintes des consommateurs ou en réalisant des enquêtes auprès des consommateurs). Le quatrième, qui est le dernier facteur, porte sur « l'échelle sur laquelle les résultats sont rapportés ». Les résultats peuvent être rapportés sur différentes échelles : au niveau "champ", au niveau "enregistrement", sur une échelle de satisfaction des consommateurs, sur l'échelle de "six sigma" ou encore sur une échelle des coûts d'une mauvaise qualité de données. Le cadre proposé par Redman formalise différentes particularités afin de choisir le bon endroit pour évaluer la qualité de données tout en définissant un scope de données pour le processus d'évaluation. Le but étant de satisfaire les utilisateurs finaux des données. Il propose un ensemble de techniques pour mesurer l'exactitude parmi lesquelles on trouve la comparaison avec les données du monde réel sans pour autant préciser comment obtenir ces données de référence.

Un autre travail réalisé par Taleb et al. [Taleb, 16] dans lequel ils proposent un système d'évaluation de l'exactitude, de l'exhaustivité et de la cohérence de données en Big Data en se basant principalement sur l'échantillonnage de données. Le principe adopté consiste à créer un ensemble d'échantillons, sans replacement, à partir de l'ensemble original des données. Ensuite, à partir de chaque échantillon créé, générer un ensemble d'échantillons en utilisant la technique de ré-échantillonnage BLB (Bag of Little Bootstraps) initialement introduite dans [Kleiner, 14]. Sur chaque échantillon ainsi généré, un processus de profilage de données est appliqué pour extraire des informations descriptives des données

telles que la description du format de données, les différents attributs, leurs types et leurs valeurs, les éventuelles contraintes, les plages des valeurs autorisées, etc. Toutes ces informations obtenues grâce au processus de profilage sont ensuite utilisées pour sélectionner les métriques appropriées pour chaque dimension avant de procéder à leur évaluation.

Concernant l'évaluation de la qualité des données non-structurées, les auteurs de [Immonen, 15] s'intéressent à l'évaluation des données collectées sur des réseaux sociaux grâce à l'intégration d'un système de gestion des métadonnées dans une architecture Big Data. Dans leur approche, les auteurs identifient cinq groupes de métadonnées : « les métadonnées de navigation » permettant de déterminer l'emplacement de chaque jeu de données ; « les métadonnées de processus » décrivant la source et les traitements effectués sur chaque jeu de données ; « les métadonnées descriptives » qui sont constituées de "métadonnées métier", décrivant la signification d'un jeu de données, et de "métadonnées techniques" fournissant les informations techniques sur le jeu de données comme la taille des données, la description du contenu, le créateur des données, le type et le format du contenu, etc. ; « les métadonnées de qualité » comprenant les dimensions et les métriques décrivant la qualité de données ; et enfin, « les métadonnées administratives » décrivant le fournisseur de données, les licences applicables et les droits d'accès sur les jeux de données, le titulaire du droit d'auteur et l'indicateur du niveau de confidentialité des données, etc. L'utilisation des métadonnées pour évaluer la qualité des données non-structurées semble être une bonne solution notamment lorsqu'il est difficile, voire impossible, de comparer ces données avec des données représentatives du monde réel. Cependant, la gestion des métadonnées peut être un processus très coûteux et complexe, en particulier pour les métadonnées de qualité. Le volume et la vélocité élevés en Big Data sont de véritables défis à relever. L'utilisation des métadonnées en conjonction avec d'autres techniques de comparaison avec des données correctes nous semble être plus efficace.

Enfin, les auteurs de [Mylavarapu, 19] se sont focalisés sur la même problématique que la nôtre, à savoir l'identification explicite des données de référence. Dans ce travail, les auteurs proposent un outil d'évaluation de l'exactitude de données basé sur la collection et la construction d'un ensemble de données en trois phases : la formation, le couplage d'enregistrements et l'évaluation de l'exactitude. L'idée est de pouvoir identifier les données de référence qui serviront ensuite de base au processus de comparaison. Dans leur approche, ils utilisent des techniques de Machine Learning pour choisir, parmi les jeux de données déjà présents dans le lac de données d'une organisation, le jeu de données le plus proche qu'ils jugent correct. Cette hypothèse peut être correcte si l'on garantit que les données présentes dans le lac de données sont correctes et à jour, ce qui n'est généralement pas le cas ; les nouvelles données recueillies peuvent même être de meilleure qualité dans certains cas. De plus, les informations correctes peuvent exister dans différents ensembles de données, pas seulement un seul comme ils le supposent. Dans ce cas, il faudra gérer de nombreux aspects comme l'hétérogénéité de schémas, le choix du jeu de données offrant la meilleure qualité pour un périmètre de données particulier, le manque de correspondance pour certaines données, etc. De plus, l'utilisation de l'intégration de mots "Word2Vec" de Google comme base pour la formation des données peut ralentir le processus d'évaluation. L'idée d'utiliser la technique de "word embedding" pour déterminer le jeu de données le plus proche reste logique mais il faudra tester ses performances dans un environnement qui héberge de très gros volumes de données.

4.3. Exactitude de données dans la littérature

4.3.1. Définition

Dans la littérature on trouve une multitude de définitions pour l'exactitude de données. Chaque définition traite des aspects relatifs au contexte dans lequel elle a été donnée. Pour Ballou et Pazer [Ballou, 85], les données sont "exactes" lorsque les valeurs de données stockées dans une base de données correspondent à des valeurs réelles. Wang et Strong [Wang, 96] définissent l'exactitude comme étant la mesure dans laquelle les données sont correctes, fiables et certifiées. Naumann et al. [Naumann, 99] et Pipino et al. [Pipino, 02] relient l'exactitude de données au pourcentage d'objets qui ne contiennent pas d'erreurs dans les données comme les fautes d'orthographe, les valeurs en dehors de la plage autorisée, etc. Loshin [Loshin, 13] précise que l'exactitude fait référence au degré auquel les valeurs des données sont correctes. Plusieurs travaux ([Redman, 96], [Scannapieco, 05], [Wand, 96] et [Batini, 06]) définissent l'exactitude comme étant une mesure de la proximité d'une valeur de donnée v à une autre valeur v' considérée comme correcte.

La norme ISO/IEC 25012 [ISO-IEC 25012, 06] ainsi que de nombreuses études ([Scannapieco, 05], [Batini, 06] et [Batini, 09]) distinguent entre l'exactitude syntaxique et l'exactitude sémantique. Chacune d'elles présente un aspect particulier et possède ses propres métriques. L'exactitude syntaxique est définie comme étant la proximité des valeurs de données à un ensemble de valeurs définies dans un domaine considéré comme étant syntaxiquement correct. Elle concerne la structure de données [Shanks, 99] et exprime le degré de non-erreur (free-of-error) de syntaxe des données. L'exactitude sémantique, quant à elle, est définie comme étant la proximité des valeurs de données à un ensemble de valeurs définies dans un domaine considéré comme étant sémantiquement correct. Elle représente le degré de validité des données [Wang, 96], [Pipino, 02] et décrit la mesure dans laquelle les données représentent les états du monde réel [Shanks, 99]. Même si elles divergent sur quelques particularités, toutes ces définitions impliquent de façon implicite ou explicite une comparaison entre les données d'un système et le monde réel qui peut être représenté par des données de référence considérées comme correctes. Par conséquent, nous adoptons la définition suivante : « *L'exactitude reflète le degré auquel les données d'un système d'information représentent le monde réel. Plus formellement, soient v la valeur d'une donnée dans un système d'information et v' une valeur de référence considérée comme correcte ; l'exactitude de v est mesurée par le degré de similarité entre v et v'.* ».

Que ce soit pour les données structurées ou semi-structurées, la comparaison des valeurs de données à évaluer avec celles correspondantes du monde réel nous permet de déduire le degré d'exactitude des informations. Toutefois, cette définition ne nous semble pas facilement implémentable aux données non-structurées comme des fichiers avec du texte libre, des fichiers multimédias (image, audio, vidéo), des données collectées sur des réseaux sociaux, etc. Certes, les données non-structurées peuvent contenir des informations qui peuvent être comparées avec le monde réel (la photo d'une personne, des informations sur un objet, une information dans l'histoire, une formule mathématique, etc.) mais le problème réside dans les informations qui ne concernent que les personnes qui les ont données (des impressions personnelles, des avis autour d'un sujet, des intentions, des rapports, etc.). Les données non-structurées sont plus complexes à évaluer et ne peuvent être évaluées de la même façon que les données structurées et semi-structurées. Si on prend l'exemple d'un article scientifique, nous ne pouvons pas évaluer sa qualité

en analysant son contenu ou en comparant les informations qu'il contient avec des données de référence. Il faut plutôt analyser des données qui lui sont attachées telles que l'importance de la revue où il a été publié, la réputation des auteurs et leurs affiliations institutionnelles, le type de l'éditeur (société savante, éditeur commercial, …), la source de l'article (revue à comité de lecture, article non publié,…), etc. Ainsi, l'évaluation de l'exactitude des données non-structurées sera plus pertinente si elle porte sur des métadonnées liées aux données plutôt que sur les données elles-mêmes. Dans [Immonen, 15], confirmant notre hypothèse, les auteurs présentent une solution pour évaluer la qualité des données collectées des réseaux sociaux en intégrant un système de gestion des métadonnées dans le cycle de vie des données Big Data.

4.3.2. Données de référence

Selon Redman [Redman, 05], il est impossible de dire par examen direct si une valeur de donnée est correcte ; toutes les mesures de l'exactitude de données doivent obligatoirement faire référence aux connaissances humaines, à d'autres données de référence ou au monde réel. La comparaison des données à évaluer avec les valeurs du monde réel rend la mesure de leur exactitude complexe et coûteuse car, très souvent, ces valeurs réelles sont inconnues [Dasu, 03], [Fugini, 02] ou sont hypothétiques, réellement indisponibles [Motro, 98]. Le degré de complexité change selon le type de l'exactitude à mesurer. L'exactitude syntaxique est généralement plus simple que l'exactitude sémantique. En effet, elle peut être vérifiée en comparant les valeurs de données avec des dictionnaires de référence tels que les dictionnaires de noms, les listes d'adresses, les dictionnaires de domaines (listes de catégories de produits, de catégories commerciales,…), les plages de valeurs, etc. En revanche, l'exactitude sémantique est plus complexe à quantifier car les termes de comparaison doivent être dérivés du monde réel. Un moyen systématique pour vérifier l'exactitude sémantique lorsque plusieurs sources de données sont disponibles consiste à comparer les informations relatives à la même instance stockées dans différents endroits. Un processus typique de vérification de l'exactitude sémantique comprend deux phases [Fugini, 02] : une phase de recherche et une phase de mise en correspondance. La phase de recherche consiste à identifier les instances concordantes. La phase de mise en correspondance consiste à prendre une décision sur la correspondance, la non-correspondance ou la correspondance possible. Ensuite, différents critères peuvent être appliqués pour effectuer la comparaison. En pratique, la comparaison se fait avec des données collectées depuis une source de référence considérée comme suffisamment fiable [Missier, 03]. Quand plusieurs sources de données fournissent les mêmes types d'informations, celles les plus fiables peuvent être considérées comme références de données pour effectuer la comparaison. La fiabilité des sources de données peut être déterminée grâce à la confiance et la réputation du fournisseur des informations. D'autres stratégies consistent à prendre en compte d'autres facteurs de qualité pour déterminer la source de données la plus fiable, par exemple, la cohérence de données [Redman, 96]. Dans certains cas, le recours à la décision d'un analyste de données peut être requis pour valoriser l'exactitude sémantique de données.

Par ailleurs, lors de la collection minutieuse de données, rien ne garantit que les informations collectées soient exactes. Aujourd'hui, pour vérifier l'exactitude des informations fournies par les consommateurs d'un service, de nouvelles méthodes sont utilisées comme l'envoi automatique d'un code secret par courrier pour vérifier une adresse postale, par courriel pour vérifier une adresse email, par SMS ou appel vocal pour vérifier un numéro de téléphone, etc. Ces méthodes, bien qu'elles soient efficaces, ne peuvent être appliquées pour estimer l'exactitude de données déjà collectées vu leurs coûts trop élevés et

le temps qu'elles peuvent prendre. Elles ne sont pas, non plus, adaptées pour vérifier l'actualité et, par conséquent, l'exactitude des informations.

Les données de référence peuvent être obtenues à travers différents mécanismes tels que :

- L'identification, si ces données existent. Lors du processus d'évaluation, à chaque source de données peut être attribué un niveau de fiabilité déterminé en fonction de la confiance et de la réputation du fournisseur des informations, estimé par des experts de données ou calculé moyennant des critères de qualité comme la cohérence de données [Redman, 96].
- La collection, si ces données n'existent pas. Certains dictionnaires de référence tels que les adresses (codes postaux, villes, rues, ...), les catalogues de produits, les listes des noms et prénoms, les valeurs possibles pour certains champs (diplômes, métiers, activités professionnelles,...), etc., peuvent être collectés indépendamment des données à évaluer pour servir de données de référence pour détecter les inexactitudes syntaxiques. Les informations métier peuvent également être collectées pour mettre à jour les données ou compléter des valeurs manquantes.
- La correction, si ces données existent avec une mauvaise qualité. Dans certains cas, les données de référence peuvent être construites en améliorant la qualité de certaines parties des données existantes. Un processus d'amélioration de la qualité de données peut être appliqué sur un échantillon de données représentant l'ensemble de données qui sera ensuite considéré comme base de référence pour évaluer le reste des données.

L'exactitude d'une information est donc calculée en fonction de son degré de similarité avec la réalité représentée par les données de référence. En suivant ce principe, nous considérons que ces données représentent l'élément de base pour tout processus d'évaluation de l'exactitude de données. Cependant, l'obtention des données de référence s'avère très complexe. Des considérations arbitraires peuvent souvent s'imposer telles que l'avis des experts de domaines, la confiance à l'égard des fournisseurs des informations et leurs réputations, etc. D'autres techniques plus formelles peuvent être adoptées telles que l'actualité des informations, l'exhaustivité de données, la cohérence entre les données, etc. Toutes ces notions peuvent être regroupées sous le terme "critères d'exactitude" que nous détaillons dans la section suivante.

4.3.3. Critères d'exactitude

Dans cette section, à travers des exemples, nous présentons quelques-uns des critères que les données doivent satisfaire afin de les considérer comme données de référence.

Le tout premier critère auquel on peut penser est la confiance à l'égard de la source de données. Si les données sont collectées auprès d'une source compétente dont on est "sûr" qu'elle fournit des données exactes, nous pouvons alors les considérer comme données de référence. A titre d'exemple, pour vérifier les informations d'identité d'une personne, la meilleure solution consisterait à les comparer avec les données des registres de l'état civil de son pays. Toutefois, cette option peut être plus complexe pour d'autres types de données. Par exemple, pour vérifier la validité des diplômes déclarés par une personne, l'idéal serait de confirmer ces informations auprès des institutions ayant délivré ces diplômes ce qui peut être une tâche très difficile, voire impossible. Il serait alors plus simple de vérifier ces informations auprès d'une organisation fournissant ce type de service et ayant une bonne réputation pour le faire. Un autre

exemple illustratif concerne les données des marchés financiers dont les valeurs des instruments changent en continue. Un trader, pour prendre une décision financière, se base généralement sur des données de marché qui lui sont transmises par des services internes de son organisation. Avant d'acter une transaction, les institutions financières exigent une étape essentielle qui consiste à valider les données sur lesquelles le trader s'est basé pour prendre sa décision avec des données achetées auprès d'organisations externes, telles que Bloomberg, Thomson Reuters, etc., qui sont spécialisées dans ce domaine et garantissent un rafraîchissement permanent de leurs données. Par ailleurs, en Big Data, de nombreux fournisseurs de données peuvent alimenter le lac de données d'une organisation par des informations autour d'un même sujet qui peuvent être redondantes ou même contradictoires dans certains cas. Pour déterminer la source la plus fiable, nous pouvons bien évidemment nous référer aux critères cités dans les exemples ci-dessus, si cela est possible, ou se référer à d'autres critères de qualité comme la cohérence de données.

En analysant ces exemples, nous comprenons que l'obtention des données de référence revient à ce que leurs fournisseurs répondent à un ensemble de critères, que nous appelons "critères d'exactitude", parmi lesquels on peut citer :

- **Confiance** : la confiance joue un rôle central dans l'évaluation de la qualité de l'information [Gamble, 11]. Elle reflète la fiabilité du fournisseur. Nous définissons une source de confiance comme étant une source de données compétente dans son domaine dans lequel elle peut fournir des données exactes.
- **Réputation** : nous définissons la réputation comme étant une mesure qui reflète l'opinion publique sur une source d'information quant à la fiabilité des données qu'elle fournit. Cette valeur évolue dans le temps en fonction des expériences passées avec la source de données. Dans le domaine de la sécurité informatique, différentes approches existent pour construire des modèles de confiance, parmi lesquels on trouve ceux à base de réputation. Ces modèles tiennent en considération les interactions et les expériences passées entre les entités [Hussain, 07].
- **Actualité** : l'actualité, ou la fraîcheur, de données peut être un critère fondamental dans certains contextes comme illustré dans l'exemple précédent. Pour Fox et al. [Fox, 94], une donnée est dite actuelle, ou à jour, à un instant t si elle est correcte à cet instant t. Une donnée est obsolète à l'instant t si elle est incorrecte à t mais était correcte à un moment quelconque avant t. L'actualité peut être évaluée au travers des mécanismes mis en place par le fournisseur de données pour mettre et garder à jour ses données.
- **Cohérence** : la cohérence de données représente le degré auquel l'information possède des attributs cohérents et sans contradiction avec d'autres informations dans le système. Elle indique si la relation logique entre les données corrélées est correcte et complète [Cai, 15]. Nous pouvons donc utiliser la cohérence comme critère qui justifie l'exactitude de données. Tout comme l'actualité, la cohérence peut être estimée au travers des mécanismes mis en place par le fournisseur de données pour synchroniser et assurer l'intégrité de ses données.

Les critères d'exactitude présentés jusqu'ici peuvent être utilisés comme indicateurs déterminant les sources de données de référence. La liste n'est pas exhaustive et le choix des critères d'exactitude est en forte dépendance avec le contexte d'application. Dans plusieurs cas, un seul critère ne serait pas suffisant et la combinaison de deux ou plusieurs critères d'exactitude serait nécessaire. Si on reprend l'exemple sur les données des marchés financiers, il se peut que deux vendeurs de données divergent sur une donnée particulière ce qui nécessite, en plus de l'actualité, d'introduire d'autres critères d'exactitude comme la confiance et/ou la réputation.

Par ailleurs, un critère d'exactitude peut être mesuré de différentes manières selon le contexte d'utilisation. L'objectif est de calculer le degré auquel une source de données satisfait le critère d'exactitude. Cette mesure peut être représentée par :

- Une valeur numérique calculée grâce à une formule mathématique appliquée à différents paramètres tels que la présence et le type d'un certificat de confiance, le protocole d'échange de données, le degré de cohérence de données, les mécanismes et les processus d'actualisation, de validation et de protection de données, l'historique des échanges avec la source, etc..
- Une valeur booléenne, ou binaire, qui montre si la source satisfait ou pas le critère d'exactitude. Cette valeur peut être calculée grâce à un seuil définissant la valeur minimale qu'une source doit atteindre.
- Une valeur dans une échelle (de 1 à 5 par exemple) qui représente le degré de satisfaction basé sur l'historique des échanges avec la source de données.
- Etc.

Dans le cas où plusieurs critères d'exactitude sont mutualisés pour déterminer la source des données de référence, un ordonnancement des critères par ordre d'importance serait obligatoire. Cela implique la nécessité d'assigner à chaque critère un "poids" permettant de résoudre les éventuels conflits. Pour ce faire, la normalisation des valeurs de mesure des différents critères est nécessaire. Par exemple, si pour un cas donné, on utilise trois critères d'exactitude, la confiance représentée par une valeur binaire (0 ou 1), la réputation représentée sur une échelle (de 1 à 5) et la cohérence représentée par un pourcentage (entre 0% et 100%), la mutualisation des trois critères nécessite la normalisation des valeurs grâce à une technique mathématique telle que "Min-Max Scaling" définie par la formule suivante :

$$\mathrm{X_{normal}} = \frac{\mathrm{X} - \mathrm{X_{min}}}{\mathrm{X_{max}} - \mathrm{X_{min}}} \qquad (4.1)$$

Avec :

- X représente la valeur du critère à normaliser.
- X_{min} représente la plus petite valeur que le critère à normaliser peut avoir.
- X_{max} représente la plus grande valeur que le critère à normaliser peut avoir.

Cette formule nous permet de transformer des valeurs ayant des unités hétérogènes en valeurs comprises dans un intervalle [0, 1] ce qui nous permet d'appliquer des études comparatives et analytiques à différents critères d'exactitude.

4.3.4. Granularité

De manière générale, la granularité désigne le niveau de détails d'une information. Par exemple, une adresse postale peut être enregistrée avec une granularité grossière dans un seul champ, comme elle peut être enregistrée avec une granularité plus fine sous forme de multiples champs comme le numéro de la rue, le code postal, la ville, etc. Les techniques d'évaluation de l'exactitude dépendent de la granularité des données à évaluer. Nous désignons par "unité de données" l'ensemble de données appartenant à un niveau de granularité. Il s'agit de l'élément de base sur lequel seront appliquées les opérations d'évaluation de l'exactitude. En bases de données relationnelles, une table est composée de plusieurs colonnes et les données sont enregistrées sous forme de tuples. Chaque tuple est considéré comme un

enregistrement représentant un objet du monde réel et composé de plusieurs cellules. Dans les bases de données orientées documents, les données sont enregistrées sous forme d'un ensemble de collections formant des objets JSON. Chaque collection est composée d'un ensemble de couples clé/valeur où la valeur peut être de type simple (chaîne de caractère, numérique, booléenne, date, etc.), de type tableau d'éléments ou de type objet JSON.

Nous pouvons ainsi distinguer entre trois niveaux de granularité. Le niveau le plus fin porte sur les valeurs ; dans ce niveau l'exactitude est obtenue en évaluant chaque valeur indépendamment des autres valeurs. Ce niveau correspond aux cellules d'une table relationnelle et aux valeurs de types simples des documents JSON. Une approche d'évaluation de l'exactitude basée sur ce niveau est très coûteuse entraînant une surcharge de calcul mais elle reste la plus précise. Le niveau suivant porte sur les objets ; il correspond aux enregistrements des tables relationnelles et aux collections dans des documents JSON. Ensuite, le niveau le plus grossier porte sur des ensembles de données comme par exemple des tables de bases de données ou des documents JSON regroupant plusieurs collections. L'inconvénient de cette approche c'est qu'elle ne montre pas où les inexactitudes sont concentrées (des cellules particulières, des champs particuliers, certains enregistrements, etc.) mais présente l'avantage de rapidité des calculs.

4.3.5. Métriques

L'évaluation de la qualité de données peut être quantitative ou qualitative [Immonen, 15]. L'évaluation quantitative est un processus systématique et formel. Elle s'appuie sur la connaissance existante d'une organisation et applique des méthodes formelles pour atteindre les valeurs des métriques dites objectives. Les résultats de l'évaluation quantitative sont donc objectifs et plus concrets que dans le cas d'une évaluation qualitative. Cette dernière repose sur des métriques, dites subjectives, mesurant les perceptions et les expériences des parties prenantes. Elles sont généralement effectuées par des utilisateurs ou des administrateurs d'informations [Pipino, 02] [Batini, 09]. Il n'y a pas d'entente dans la littérature sur les types des métriques pour mesurer l'exactitude de données. On en trouve plusieurs parmi lesquelles on peut citer :

- **Métrique booléenne** : si l'unité de données à évaluer représente correctement l'objet réel, alors l'exactitude prend la valeur "vrai" sinon elle prend la valeur "faux".
- **Métrique en degré** : cette métrique est calculée en divisant le nombre des unités de données correctes sur le nombre total des unités de données. Ceci permet d'exprimer le degré de confiance en ces données.
- **Métrique d'écart** : il s'agit d'une valeur numérique qui capture la distance entre une unité de données du système et une référence qui représente le monde réel. Généralement, cette métrique est calculée grâce à la distance entre les objets. Plus la distance est grande, plus les objets ne se ressemblent pas.

Pour mesurer quantitativement l'exactitude de données, des fonctions d'agrégation doivent être mises en place. Soient S un échantillon de n unités de données à évaluer et A_i la valeur de l'exactitude de l'$i^{ème}$ élément dans S. En pratique, déterminer ce qui constitue une unité de données et ce qui est une erreur nécessite un ensemble de critères clairement définis [Pipino, 02] qui dépendent du contexte de chaque projet. A titre d'exemple, il est possible qu'un caractère incorrect dans une chaîne de caractères

soit tolérable dans un cas mais pas dans un autre. Inspirés de [Peralta, 06], [Pipino, 02], [Loshin, 01] et [Batini, 09], nous proposons deux fonctions d'agrégation typiques :

- **Ratio** : cette technique mesure le rapport entre le nombre d'unités de données correctes dans l'échantillon, divisé par la cardinalité de l'échantillon.
  - Si l'on considère que l'exactitude des unités de données est exprimée à l'aide d'une mesure booléenne, c'est-à-dire $A_i \in \{0,1\}\ avec\ 1 \leq i \leq n$, alors l'exactitude de S est calculée comme suit :

$$\text{Accuracy (S)} = \frac{|\{A_i\}, A_i = 1|}{n} = \frac{\sum_{i=1}^{n} A_i}{n} \qquad (4.2)$$

    $|\{A_i\}, A_i = 1|$ dénote la cardinalité des unités des données ayant une exactitude correcte.
  - Si l'on considère que l'exactitude des unités de données est exprimée à l'aide d'une mesure en degré ou en distance, c'est-à-dire $A_i \in [0, 1]$ avec $1 \leq i \leq n$, alors il faudra considérer un seuil θ à partir duquel la donnée est considérée comme correcte. Dans ce cas, l'exactitude de S est calculée comme suit :

$$\text{Accuracy (S)} = \frac{|\{A_i\}, A_i \geq \theta|}{n} \qquad (4.3)$$

    $|\{A_i\}, A_i \geq \theta|$ dénote la cardinalité des unités des données ayant une exactitude supérieure ou égale à θ.
- **Moyenne** : cette technique permet de mesurer la moyenne des unités de données correctes. Quelle que soit la métrique utilisée, l'exactitude de S est calculée comme suit :

$$\text{Accuracy (S)} = \frac{\sum_{i=1}^{n} A_i}{n} \qquad (4.4)$$

  - Si l'on considère que les unités des données n'ont pas toutes la même importance, nous pouvons assigner à chaque unité un poids w_i et calculer l'exactitude de S avec la formule suivante :

$$\text{Accuracy (S)} = \frac{\sum_{i=1}^{n} w_i A_i}{\sum_{i=1}^{n} w_i} \qquad (4.5)$$

4.3.6. Techniques de comparaison de données

La comparaison de données est un processus qui paraît simple, mais dans le contexte Big Data, elle demande la mise en place de certaines techniques et l'implémentation de certains concepts complexes. Nous avons déjà abordé dans le chapitre 2 certains de ces concepts et techniques : correspondance de schémas, couplage d'enregistrements, etc. Dans le reste de cette section, nous allons présenter brièvement d'autres techniques à savoir "l'échantillonnage de données en Big Data" et "les techniques de mesures de similarité de données".

4.3.6.1. Echantillonnage de données en Big Data

La taille des données est un enjeu essentiel en Big Data. Lorsque cette taille dépasse les capacités de mémoire et de traitement des machines, les processus nécessitant le chargement de la totalité ou de gros volumes de données deviennent prohibitifs ou impossibles. L'évolution des ressources et la mise en place de nouvelles architectures de stockage et de traitement de données sont devenues une nécessité incontournable pour saisir et traiter ces gros volumes de données. Pour satisfaire ces besoins, deux types de dimensionnement (Scaling, en anglais) existent. La première solution, appelée « Scale-Up », consiste à

augmenter la taille de la mémoire, la taille de stockage et la puissance des processeurs d'une machine. Cette solution atteint rapidement ses limites si le volume de données à traiter dépasse la nouvelle configuration des machines de stockage et de traitement. La deuxième solution, appelée « Scale-Out », s'avère plus intéressante. Elle consiste à distribuer le stockage et le traitement sur plusieurs machines. Cette deuxième solution, bien adaptée aux environnements Big Data, s'est épanouie avec l'émergence de Hadoop et son écosystème ainsi que le Cloud Computing. Une troisième solution très abordée dans la littérature et bien adaptée à notre problématique est l'échantillonnage de données. En effet, le besoin d'une réponse rapide est parfois plus important qu'une réponse précise notamment pour le cas d'évaluation de l'exactitude de données. Nous pouvons considérer l'échantillonnage, qu'on qualifiera de « Scale-Down », comme une alternative pour traiter des données volumineuses. Cette technique s'avère extrêmement utile pour rendre la Big Data exploitable pour l'analyse [Pyne, 16]. Lorsque l'on souhaite analyser des ensembles de données en vue d'évaluer leur qualité, nous pouvons nous contenter de la sélection et de l'analyse d'un échantillon représentatif de toutes les unités de données. Pour certains types de problèmes, l'analyse d'un échantillon de données donne des résultats aussi bons que l'analyse de l'ensemble de données [Dean, 14], mais pour des cas particuliers, notamment l'analyse des gros volumes de données, l'échantillonnage parait être la solution la plus adaptée [Dasu, 03], [Motro, 98], [Prajapati, 13].

Pour que les résultats de l'analyse sur l'échantillon puissent être extrapolés aux ensembles de données faisant l'objet de l'évaluation, il est indispensable que cette analyse soit conduite selon des règles bien définies et que les estimations conduisant à ces extrapolations soient conformes à la procédure d'échantillonnage utilisée. L'échantillon choisi doit être le plus représentatif possible des données à évaluer. Le degré de correspondance entre l'information recueillie et ce que nous apprendrait un recensement dépend en grande partie de la façon dont l'échantillon a été choisi. La théorie moderne de l'échantillonnage nous propose une distinction fondamentale entre des échantillons probabilistes et des échantillons empiriques :

- **échantillons probabilistes** : on y trouve principalement l'échantillonnage aléatoire et simple, l'échantillonnage stratifié, l'échantillonnage par grappes, l'échantillonnage par degrés et l'échantillonnage systématique. Le graphe de la Figure 4.1 présente une classification des méthodes d'échantillonnage probabiliste selon le type de données (homogène ou hétérogène).
- **échantillons empiriques** : on y trouve principalement l'échantillonnage accidentel (ou de convenance), l'échantillonnage à priori, l'échantillonnage "boule de neige", l'échantillonnage par Quotas, etc. Ce dernier étant le plus utilisé des échantillons empiriques.

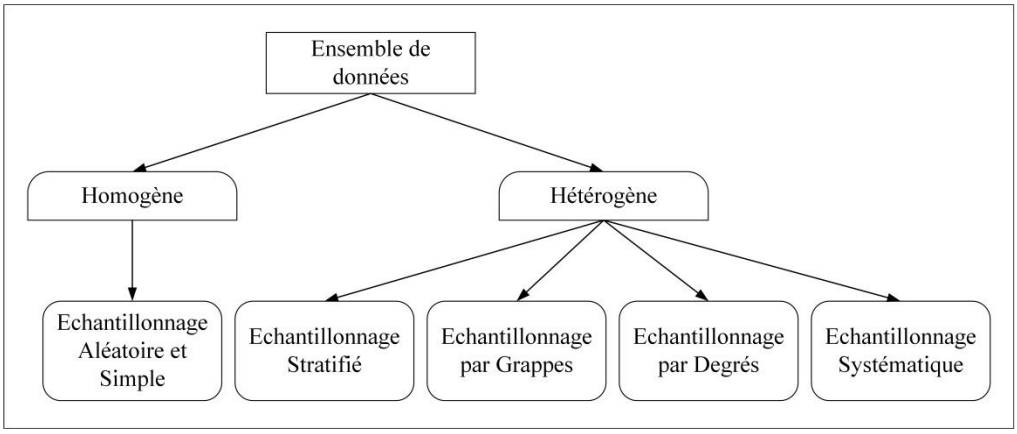

Figure 4.1. Classification des méthodes d'échantillonnage probabiliste

Dans la pratique, les échantillons empiriques sont moins utilisés que les échantillons probabilistes. Ces derniers sont complémentaires et peuvent être étendus pour répondre à des besoins plus spécifiques. Les principales techniques de l'échantillonnage probabiliste sont :

- **Echantillonnage aléatoire et simple** : l'échantillonnage est aléatoire si toutes les unités de données ont la même chance de faire partie de l'échantillon. Il est simple si les prélèvements des unités de données sont réalisés indépendamment les unes des autres. Pour prélever un échantillon aléatoire et simple, voici les étapes à suivre :
  - Tout d'abord, constituer la base de sondage qui correspond à la liste complète et sans répétition des unités de données.
  - Ensuite, numéroter ces éléments de 1 à N.
  - Enfin, procéder, à l'aide d'une table de nombres aléatoires ou d'un générateur de nombres aléatoires, à la sélection des unités différentes qui constitueront l'échantillon.
- **Echantillonnage stratifié** : c'est une technique qui consiste à subdiviser un ensemble de données hétérogènes de cardinalité N, en P sous-ensembles homogènes. Chaque sous-ensemble est une "strate" ayant une cardinalité N_i tel que, $N = N_1 + N_2 + \cdots + N_p$. Un échantillon de cardinalité n_i est prélevé indépendamment au sein de chacune des strates en appliquant une méthode d'échantillonnage adaptée (par exemple, un échantillonnage aléatoire et simple à l'intérieur de chaque strate). Pour la répartition des n unités de données de l'échantillon dans les différentes strates, on utilise une répartition proportionnelle qui consiste à conserver la même fraction d'échantillonnage dans chaque strate ou à appliquer un poids d'importance de chaque strate. Désignons par w_i le poids de la $i^{ème}$ strate ($w_i = \frac{N_i}{N}$) et par f la fraction de sondage constante ($f = \frac{n}{N}$). Le nombre d'unités de données à choisir dans cette strate est : $n_i = w_i * n = f * N_i$
- **Echantillonnage par grappes** : dans l'échantillonnage par grappes, on tire au hasard des grappes ou familles de données (bases de données, répertoire de fichiers, etc.), et on examine toutes les unités de données au sein de chaque grappe (par exemple, on sélectionne au hasard des dossiers et on analyse ensuite le contenu des dossiers sélectionnés). La méthode est d'autant meilleure que les grappes se ressemblent et que les données d'une même grappe sont différentes, contrairement aux strates. On divise la population en sous-groupes appelés grappes (ou clusters). Les grappes ont le même profil et la variance d'une grappe à l'autre est faible. On sélectionne par la suite un échantillon aléatoire de grappes et non pas un échantillon aléatoire à l'intérieur de chaque grappe. La taille de l'échantillon final est égale à la somme des tailles des grappes sélectionnées.
- **Echantillonnage par degrés** : ce type d'échantillonnage regroupe toute une série de plans d'échantillonnage caractérisés par un système ramifié et hiérarchisé d'unités. On procède à un premier tirage pour construire un premier échantillon ; chaque élément de cet échantillon de 1er degré contient un ensemble d'éléments. On procède ensuite à un deuxième tirage au sein de chaque élément pour construire les échantillons de 2ème degré et ainsi de suite jusqu'à arriver aux éléments que l'on souhaite analyser. Le mode de sélection pouvant varier d'un niveau à l'autre.
- **Echantillonnage systématique** : cette technique consiste à prélever des unités d'échantillonnage situées à intervalles égaux. Le choix de la première unité de données détermine la composition de tout l'échantillon. Si N est la cardinalité de l'ensemble de données à échantillonner en un échantillon de taille n, l'intervalle entre deux unités successives à sélectionner est donné par $k = \frac{N}{n}$ (arrondi à l'entier le plus proche). Connaissant k, on choisit un nombre aléatoire i compris entre 1 et k. Le rang des unités sélectionnées est alors $i, i + 2k, i + 3k, \ldots$. L'échantillonnage

systématique est facile à préparer et réduit le temps consacré à la localisation des unités à sélectionner.

Plusieurs techniques peuvent être utilisées conjointement pour créer un échantillon efficace. Quelles que soient les techniques utilisées, toutes les unités de données doivent avoir la même chance d'être sélectionnées dans l'échantillon. En outre, pour connaitre la taille de l'échantillon, il faudra connaitre au préalable la taille des données à échantillonner ce qui n'est pas facile à obtenir dans une application Big Data. Pour résoudre ces problèmes, il existe une approche efficace appelée "Reservoir Sampling" introduite initialement par Vitter [Vitter, 85] en 1985. L'échantillonnage de réservoir est une famille d'algorithmes randomisés permettant de choisir au hasard un échantillon de k éléments dans un grand ensemble de données de taille n ou dans un flux de données de taille n, où n est inconnu ou difficile à connaitre. Tous les éléments ont une probabilité identique pour qu'ils soient sélectionnés dans l'échantillon. Le principe est le suivant : Soit S l'ensemble ou le flux de donner à échantillonner ; on commence par créer un échantillon des k premiers éléments qu'on appellera le Réservoir R et ensuite, par accès séquentiel sur le reste des éléments de S, on remplace aléatoirement des éléments dans R. Voici un algorithme type de la technique "Reservoir Sampling" :

```
1. Soit S l'ensemble ou le flux de données à échantillonner.
2. Créer un réservoir R vide de taille maximale k.
3. Remplir le réservoir R par les k premiers éléments de S.
4. Pour chaque élément, de la position k+1 jusqu'au dernier élément dans S,
   refaire le traitement suivant :
   4.1.     Soit i la position de l'élément en cours dans S.
   4.2.     Soit j un chiffre généré aléatoirement entre 0 et i.
   4.3.     Si j < k, alors remplacer R[j] par S[i].
```

L'avantage de cette méthode c'est qu'elle nous permet de créer un échantillon en parcourant les données une seule fois, au fur et à mesure de la création de l'échantillon, par accès séquentiel, sans connaissance préalable de la taille des données à échantillonner.

4.3.6.2. Mesures de similarité

De nombreuses méthodes existent pour mesurer la similarité entre les données. Le choix d'une méthode dépend en grande partie du type des données à comparer (chaînes de caractères, nombres, valeurs binaires, etc.) et du contexte d'utilisation (par exemple, pour un besoin particulier, nous pouvons considérer que deux très grands nombres sont similaires même s'ils ne sont pas exactement égaux).

Pour comparer les chaînes de caractères, de nombreuses méthodes existent telles que la distance de Levenshtein [Levenshtein, 66] et la distance de Jaro-Winkler [Winkler, 90]. La distance de Levenshtein, souvent appelée "distance d'édition", peut être définie comme le nombre minimum de modifications nécessaires pour convertir une chaîne de caractères en une autre. Selon le contexte de chaque projet, des ajustements peuvent être nécessaires pour adapter cette méthode à des besoins particuliers (sensibilité à la casse, caractères accentués et spéciaux, utilisation des acronymes, etc.) [Olensky, 14], [Yujian, 07]. La distance de Jaro-Winkler, quant à elle, mesure la similarité entre deux chaînes de caractères. Il s'agit d'une variante proposée par Winkler découlant de la distance de Jaro [Jaro, 89] utilisée dans le domaine du couplage d'enregistrements pour la détection des doublons. La distance de Jaro-Winkler consiste à calculer le nombre de caractères que les chaînes ont en commun. De nombreuses autres méthodes existent dans ces thématiques telles que la similarité Cosine, q-Gram, Damerau-Levenshtein, etc.

Pour les données numériques, la similarité peut être calculée en tant que différence entre les valeurs [Shankaranarayanan, 03] en tenant en considération un seuil à partir duquel on peut considérer que deux nombres sont similaires. Le choix du seuil dépend du contexte de comparaison et de la grandeur des nombres (une différence entre deux petits nombres n'a pas le même effet que celle entre deux très grands nombres). D'autres types, comme les dates et les coordonnées géographiques, peuvent suivre ce même principe puisqu'ils sont convertibles en nombre en récupérant le timestamp des dates et les latitude/longitude des coordonnées géographiques. Enfin, pour les données booléennes et les caractères, la similarité implique une égalité exacte entre les données.

4.4. Evaluation de l'exactitude de données en Big Data

Dans cette section, nous présentons notre solution pour évaluer l'exactitude de données dans un contexte Big Data. Nous exposons d'abord un modèle pour comprendre les principales étapes du processus d'évaluation ainsi que l'architecture mettant en relation les différents modules de la solution. Ensuite, pour prouver notre concept, nous présentons une étude de cas ainsi que les étapes d'implémentation et de validation de notre approche. Enfin, pour revenir sur l'objectif de ce chapitre, nous allons présenter notre analyse afin d'en déduire des conclusions.

4.4.1. Cadre d'évaluation de l'exactitude de données

Dans [Talha, 20-b], nous avons proposé un processus d'évaluation de l'exactitude de données, illustré dans la Figure 4.2, qui consiste en cinq étapes :

1. **Master Data Set** : la toute première étape consiste à collecter des données auprès des fournisseurs internes ou externes d'une organisation. Ces données sont enregistrées dans notre lac de données (Data Lake) qui constitue notre "Master Data Set" dans son état brut. La collection des données se fait grâce à des outils d'ingestion permettant d'ouvrir des canaux sécurisés entre le Data Lake et les sources de données (bases de données relationnelles, des répertoires de données synchronisés, des outils de messageries, …). Comme expliqué dans le chapitre 1, de nombreux outils de l'écosystème Hadoop tels que Apache Sqoop, Apache Flume ou encore Apache Kafka peuvent être utilisés dans cette phase d'ingestion de données.
2. **Golden Data Set** : cette étape nécessite une étude préliminaire réalisée au sein de l'organisation souhaitant adopter notre solution. Le but est de déterminer les critères d'exactitude qui caractérisent au mieux les données de "bonne qualité". Pour notre étude de cas, nous avons choisi arbitrairement trois critères d'exactitude : la Confiance **T** (pour Trust) représentée par un pourcentage, la Réputation **R** (pour Reputation) mesurée sur une échelle de 1 à 5 et la Cohérence **C** (pour Consistency) représentée également par un pourcentage. La construction du "Golden Data Set" consiste donc à appliquer des mesures permettant d'assigner à chaque ensemble de données du Master Data Set des métadonnées reflétant les valeurs de chaque critère d'exactitude. Pour notre étude de cas, nous appliquons des algorithmes hypothétiques, reposant sur des hypothèses. Dans un cas réel, la valeur de la Confiance peut être déterminée par un expert de données ou calculée en fonction du certificat de confiance de la source de données ; la valeur de la Réputation peut être obtenue grâce à des retours d'expériences ou des questionnaires destinés aux

utilisateurs ; la valeur de la Cohérence peut être calculée en scannant et en analysant l'ensemble des données du système d'information de la source de données, etc.

3. **Mapped Data** : cette étape consiste à mettre en relation les colonnes des données à évaluer avec celles du Golden Data Set. A noter qu'une colonne peut être mappée avec zéro, une ou plusieurs colonnes des ensembles de données composant le Golden Data Set. En cas d'absence correspondance, la valeur de l'exactitude ne peut être calculée. En cas de correspondance avec une seule information existant dans un seul ensemble de données du Golden Data Set, le calcul de l'exactitude est relativement simple ; il suffit de calculer la similarité entre les deux informations. En cas de correspondance avec plusieurs informations existant dans différents ensembles de données du Golden Data Set, un conflit de correspondance existe et nécessite sa résolution grâce à un mécanisme de résolution de conflit dédié. Cette étape reste relativement simple si on se contente des données structurées ou semi-structurées. Pour les données non-structurées, il faudra penser à des solutions plus adaptées. Dans notre étude de cas, afin de simplifier le développement de notre preuve de concept, nous admettons que les données sont structurées.
4. **Reference Data** : c'est l'étape clé de notre solution. Elle consiste à identifier les données de référence en se basant sur les critères d'exactitude définis au préalable (T, R et C). La sélection de ces données de référence nécessite la mise en place d'un processus de couplage d'enregistrements, de résolution des éventuels conflits de correspondance et d'échantillonnage.
5. **Data Accuracy** : la dernière étape consiste à calculer la similarité entre les données à évaluer et les données de référence afin d'en déduire la valeur de l'exactitude.

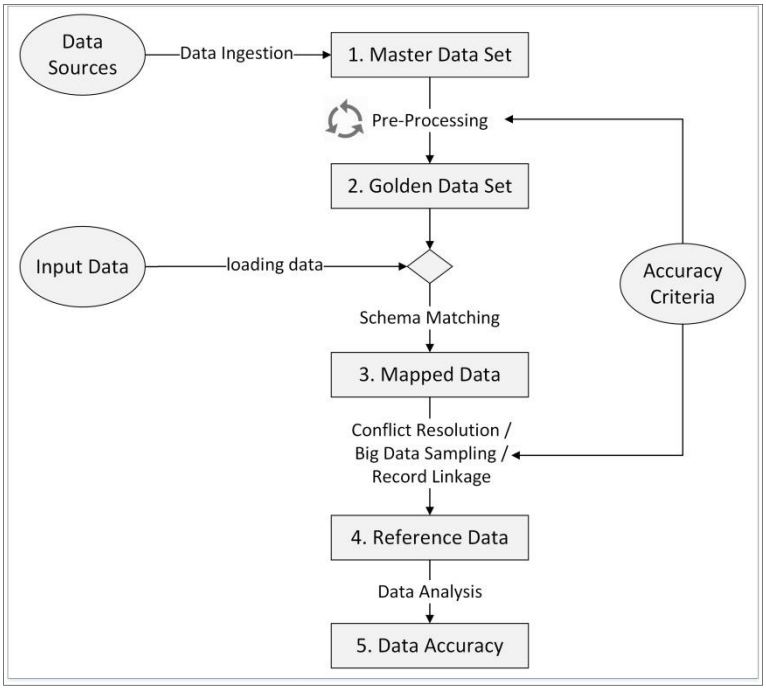

Figure 4.2. Processus d'évaluation de l'exactitude de données [Talha, 20-b]

Basés sur le modèle de la Figure 4.2, nous proposons l'architecture de notre cadre d'évaluation de l'exactitude de données dans la Figure 4.3.

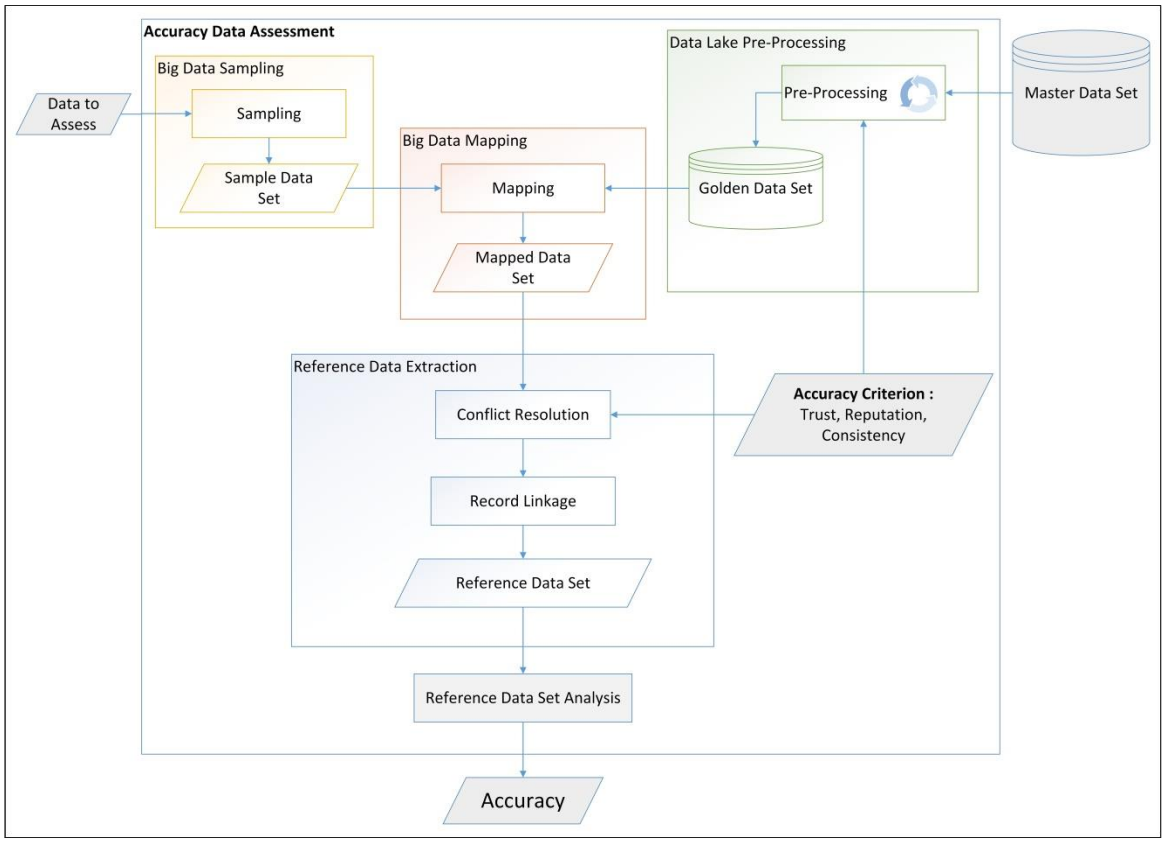

Figure 4.3. Cadre d'évaluation de l'exactitude de données en Big Data

Notre architecture met en relation différents composants et processus :

- **Big Data Sampling** : selon la volumétrie des données et le temps accordé au processus d'évaluation, la création d'un échantillon des données à évaluer peut s'avérer nécessaire. Nous proposons d'utiliser l'algorithme "Reservoir Sampling" présenté à la section 4.3.6.1 pour réaliser cet échantillonnage.
- **Data Lake Pre-processing** : ce processus consiste à construire le Golden Data Set comme expliqué à l'étape 2 du modèle de la Figure 4.2. L'objectif est d'assigner à chaque ensemble de données du Master Data Set des métadonnées concernant principalement les valeurs des critères d'exactitude. Etant donné que la fiabilité des fournisseurs de données change en fonction de l'évolution des technologies, des procédures de maintenance mises en œuvre, de l'amélioration continue, etc., les critères d'exactitude changent en conséquence, d'où la récursivité de ce processus. L'objectif est de mettre à jour de façon continue les valeurs des critères d'exactitude pour chaque ensemble de données.
- **Big Data Mapping** : ce processus consiste à réaliser un premier rapprochement entre les données à évaluer et celles présentes dans le Data Lake. L'objectif est de réaliser un premier filtrage permettant de relier les schémas des données à évaluer avec celle du Golden Data Set.
- **Reference Data Extraction** : ce processus consiste à construire les données de référence qui seront la base de calcul de l'exactitude des données à évaluer. Les données de référence doivent répondre aux critères d'exactitude pour qu'elles soient prises en considération.

- **Reference Data Set Analysis** : le dernier processus consiste à analyser les jeux de données pour mesurer la similarité entre les données à évaluer, représentées par un échantillon, et les données de référence qui répondent aux critères d'exactitude.

4.4.2. Preuve de concept

4.4.2.1. Etude de cas

Dans le cadre de nos travaux de recherche sur l'évaluation de l'exactitude de données dans le contexte Big Data, et afin de démontrer la faisabilité et la fiabilité de notre solution, nous avons mis en place une preuve de concept. Pour notre étude de cas, nous nous intéressons à l'étude des informations sur les gares ferroviaires de la région parisienne en France. Notre démarche consiste à préparer un Data Lake à partir de sources de données ouvertes "Open Data" trouvées sur Internet. Le choix du sujet est justifié par l'existence de plusieurs sources de données ouvertes offrant des données volumineuses que l'on peut utiliser gratuitement pour la recherche scientifique. Toutefois, quel que soit le contexte d'application de notre modèle, son implémentation doit être réalisable avec un minimum d'ajustement sans modifier le principe de base. Les données de notre étude sont collectées depuis cinq bases de données ouvertes :

- **Le site de la région Ile-de-France**[34] : il s'agit d'une plateforme ouverte de données publiques concernant la région Ile-de-France. La plateforme est gérée par la direction de la communication du conseil régional de l'Ile-de-France. Cette plateforme[35] offre des informations sur 1140 gares ferroviaires de la région parisienne.
- **Le site de la RATP**[36] : la Régie Autonome des Transports Parisiens (RATP) est un établissement public à caractère industriel et commercial détenu à 100% par l'état français. Il s'agit d'une régie qui assure l'exploitation, la maintenance et l'ingénierie des réseaux d'une partie des transports en commun de la ville de Paris et de sa banlieue. Cette plateforme[37] offre des informations sur 955 gares ferroviaires de la région parisienne.
- **Le site de la SNCF**[38] : la Société Nationale des Chemins de Fer (SNCF) est un établissement public à caractère industriel et commercial détenu à 100% par l'état français. Il s'agit d'une régie qui gère le transport des voyageurs et des marchandises et réalise la gestion, l'exploitation et la maintenance du réseau ferré en France. Cette plateforme[39] offre des informations sur 7702 gares ferroviaires dans toute la France.
- **Le site de l'Île-de-France Mobilité**[40] : anciennement appelée STIF (Syndicat des Transports d'Île-de-France), l'Île-de-France Mobilité est un établissement public administratif français qui représente l'autorité organisatrice des transports de la région Île-de-France. Cette plateforme[41] offre des informations sur 1140 gares ferroviaires de la région parisienne.

[34] https://data.iledefrance.fr/
[35] https://data.iledefrance.fr/explore/dataset/gares-et-stations-du-reseau-ferre-dile-de-france-par-ligne/table/
[36] https://data.ratp.fr/
[37] https://data.ratp.fr/explore/dataset/accessibilite-des-gares-et-stations-metro-et-rer-ratp/table/
[38] https://data.sncf.com/
[39] https://ressources.data.sncf.com/explore/dataset/liste-des-gares/table/
[40] https://data.iledefrance-mobilites.fr/
[41] https://data.iledefrance-mobilites.fr/explore/dataset/emplacement-des-gares-idf/table/

- **Le site du gouvernement français**[42] : il s'agit d'une plateforme ouverte de données publiques françaises développée et maintenue par Etalab[43] qui est une administration du gouvernement français placée au sein de la direction interministérielle du numérique et du système d'information et de communication de l'état. Cette plateforme[44] offre des informations sur 955 gares ferroviaires de la région parisienne.

Avant de poursuivre notre étude, une analyse des sources de données nous semble nécessaire pour bien maitriser le sujet. Suite à cette analyse, nous avons remonté les remarques suivantes :

- les données collectées du site de l'Île-De-France et celles du site Île-De-France Mobilité sont identiques. Nous avons donc considéré un seul ensemble de données dans notre Data Lake pour ces deux sources de données. Le fichier d'entrée s'appelle « dataset_idf_1140.csv ».
- les données collectées du site de la RATP et celles du site du gouvernement sont identiques. Nous avons donc considéré un seul ensemble de données dans notre Data Lake pour ces deux sources de données. Le fichier d'entrée s'appelle « dataset_ratp_955.csv ».
- les données collectées du site de la SNCF sont uniques. Le fichier d'entrée s'appelle « dataset_sncf_7702.csv ».
- Les schémas des trois ensembles de données constituant ainsi notre Master Data Set sont différents. Ils se partagent certaines informations (classées dans des colonnes avec des libellés différents) et se spécialisent pour d'autres types d'informations.
- Plusieurs colonnes sont en étroites liaisons avec le système d'information de la source de données et n'apporte pas de valeur ajoutée à notre étude de cas. Nous avons donc décidé de ne garder que les colonnes qui nous seront utiles. Les schémas des données retenus pour les trois sources de données sont représentés dans la Table 4.1.

Table 4.1. Schémas des données retenues pour les trois sources de données

dataset_idf_1140.csv		
Champ	Descriptif	Format
nom_gare	Nom de la gare	String
Geo Point	latitude, longitude	Decimal, Decimal
dataset_ratp_955.csv		
Champ	Descriptif	Format
nomptar	Nom de la gare	String
CodeINSEE	Code Insee	Integer
coord	latitude, longitude	Decimal, Decimal
AnnonceSonoreProchainPassage	Annonce Sonore Prochain Passage	Boolean
AnnonceVisuelleProchainPassage	Annonce Visuelle Prochain Passage	Boolean
AnnonceSonoreSituationsPerturbees	Annonce Sonore Situations Perturbees	Boolean
AnnonceVisuelleSituationsPerturbees	Annonce Visuelle Situations Perturbees	Boolean
dataset_sncf_7702.csv		
Champ	Descriptif	Format
LIBELLE_GARE	Nom de la gare	String
COMMUNE	la commune	String
DEPARTEMENT	le département	String
coordonnees_geographiques	latitude, longitude	Decimal, Decimal

[42] https://www.data.gouv.fr/
[43] https://www.etalab.gouv.fr/
[44] https://www.data.gouv.fr/fr/datasets/accessibilite-des-gares-et-stations-de-metro-et-rer-ratp-1/

Nous pouvons remarquer que deux types d'informations existent dans les trois sources de données (les noms et les coordonnées géographiques des gares) même si elles ne portent pas les mêmes libellés et que d'autres informations sont présentes dans une source et pas dans d'autres (les équipements des gares, les départements, …).

Par ailleurs, pour la sélection des données de référence lors du calcul de l'exactitude, nous avons considéré trois critères à savoir la confiance, la réputation et la cohérence. Bien entendu, le choix des critères d'exactitude doit être adapté selon le besoin de chaque projet. Par exemple, pour un système de trading bancaire où les données changent constamment, l'actualité des données peut être un bon critère décisif pour valoriser l'exactitude de données. Pour notre étude de cas, la confiance est représentée par un pourcentage (entre 0% et 100%), la réputation est mesurée sur une échelle (de 1 à 5) et la cohérence est représentée par un pourcentage (entre 0% et 100%).

4.4.2.2. Implémentation du concept

Les données téléchargées de ces trois sources de données ouvertes constituent notre "Master Data Set" auquel nous avons appliqué des algorithmes hypothétiques pour obtenir notre "Golden Data Set" en assignant à chaque ensemble de données des métadonnées qui reflètent le niveau de chaque critère d'exactitude. A chaque fois que l'on souhaite évaluer l'exactitude d'une fiche de données contenant des informations sur les gares ferroviaires de la région parisienne, nous faisons appel à notre processus comme expliqué ci-dessus. Nous avons implémenté notre solution en cinq étapes :

1. Mise en place de la plateforme Big Data
2. Construction du Master Data Set
3. Construction Golden Data Set
4. Mise en correspondance des données
5. Identification des données de référence et Calcul de l'exactitude

4.4.2.2.1. Mise en place de la plateforme Big Data

Nous avons développé notre solution sur une plateforme Hadoop (version 2.7.7) installée sur machine virtuelle Ubuntu 18.04.2 LTS de 64 bits ayant une mémoire vive de 8 Go et un disque SATA de 100 Go. Nous avons choisi python (version 3.6.7) comme langage de programmation et Spark (version 2.3.1) comme plateforme de traitement de données. Nous avons utilisé une base de données HBase (version 1.1.0) pour la gestion des métadonnées. Dans l'annexe A, nous fournissons le code source Python et Spark pour les différents modules de notre démonstrateur.

4.4.2.2.2. Construction du Master Data Set

Notre Data Lake est composé de trois ensembles de données construits à partir des fichiers csv téléchargés sur les sites data.iledefrance.fr, data.ratp.fr et data.sncf.com. Les données sont stockées de façon distribuée sur HDFS dans leurs états bruts (aucune modification ne leur a été attribuée). Dans l'annexe B, nous fournissons les principales commandes pour importer les données sur le Data Lake ainsi que les scripts Shell pour manipuler la plateforme Hadoop (démarrage et arrêt des services, etc.).

4.4.2.2.3. Construction du Golden Data Set

Le but de cette étape est d'étudier l'impact des critères d'exactitude sur le processus d'identification des données de référence et, par conséquent, sur le calcul de l'exactitude. Pour cela, à chaque ensemble de données nous avons attribué des valeurs hypothétiques de sorte que chaque source de données ait le meilleur score pour un critère d'exactitude parmi les trois. Cela permettra de n'exclure aucune source de données du processus expérimental de calcul de l'exactitude. Pour chacun des trois critères d'exactitude que nous avons définis :

- **Confiance** : une source de données de confiance est une source authentique gérée par une organisation gouvernementale, une institution réglementée, un corpus de connaissances ou toute autre organisation promouvant des données authentiques. Pour déterminer le degré de confiance d'une source de données, nous pouvons adopter certains critères tels quel la possession d'un certificat d'authenticité délivré par une autorité de certification, la nature de l'organisation gérant la source (gouvernemental ou privée, à but lucratif ou pas, etc.), la fréquence et les dates des dernières mises à jour, etc. Voici des valeurs arbitraires pour le critère "Confiance" :

Source de données	Degré de confiance (en pourcentage %)
Région de l'Île-de-France	91
RATP	85
SNCF	60

- **Réputation** : la réputation d'une source de données peut être une mesure de la précision avec laquelle les opinions ou les prédictions de la source correspondent à des résultats vérifiables [Delepet, 09]. La réputation peut être assignée aux ensembles de données par des utilisateurs suite à des retours d'expériences, des administrateurs de données, des analystes de données, etc., moyennant un système de notation ou toute autre solution. Voici des valeurs arbitraires pour le critère "Réputation" :

Source de données	Valeur de la réputation (sur une échelle de 1 à 5)
Région de l'Île-de-France	2.68
RATP	4.68
SNCF	3.60

- **Cohérence** : la cohérence représente le degré auquel l'information possède des attributs cohérents et sans contradiction avec d'autres informations dans un contexte d'utilisation spécifique [Rafique, 12]. Elle exige que des enregistrements multiples des valeurs des attributs d'une entité soient identiques ou très similaires dans le temps ou l'espace [Bovee, 03]. La valorisation de la cohérence peut être réalisée simplement en mesurant l'égalité entre la valeur d'une donnée et toutes ses copies dans le système. Redman considère qu'un ensemble de données cohérent mérite d'être une source de données de référence [Redman, 96]. Voici des valeurs arbitraires pour le critère "Cohérence" :

Source de données	Degré de cohérence (en pourcentage %)
Région de l'Île-de-France	47
RATP	75
SNCF	93

Par ailleurs, la normalisation des données est une transformation appliquée aux données lorsque l'incompatibilité des unités de mesures entre les variables peut affecter les résultats d'analyse. Pour notre étude de cas :

- la confiance est représentée par un pourcentage (entre 0% et 100%).

- la réputation est mesurée sur une échelle (de 1 à 5).
- la cohérence est représentée par un pourcentage (entre 0% et 100%).

La normalisation permet d'ajuster une série de valeurs représentant un ensemble de mesures en appliquant une fonction de transformation pour les rendre comparables. Pour normaliser les valeurs de nos critères d'exactitude, nous avons appliqué la technique Min-Max Scaling (formule 4.1). Pour les valeurs arbitraires précédentes, voici les valeurs normalisées :

Source de données	Confiance	Réputation	Cohérence
Région de l'Île-de-France	0.91	0.42	0.47
RATP	0.85	0.92	0.75
SNCF	0.60	0.65	0.93

Ces informations sur les niveaux relatifs à chaque critère d'exactitude pour les sources de données ainsi que d'autres informations nécessaires à l'analyse tel que le chemin vers les données dans le Data Lake, doivent être sauvegardées sous forme de métadonnées. Afin de répondre à ce besoin, nous avons choisi d'enregistrer ces informations dans une base de données HBase. Les métadonnées sont enregistrées dans une table HDFS_METADATA avec deux familles de colonnes :

- **Data_Source** qui contient deux colonnes de qualifications : **File_Path** qui représente le chemin vers le fichier de données et **File_Source** qui contient le nom de la source de données.
- **Accuracy_Criteria** qui contient trois colonnes de qualifications : **Trust**, **Reputation** et **Consistency**. Chacune de ces colonnes contient la valeur hypothétique normalisée que nous avons attribuée à chaque critère d'exactitude pour chaque source de données.

  Le contenu de notre table **HDFS_METADATA** ressemble à :

HDFS_METADATA					
ID	Data_Source		Accuracy_Criteria		
	File_Path	File_Source	Trust	Reputation	Consistency
1001	dataset_idf_1140.csv	Île-de-France	0.91	0.42	0.47
1002	dataset_ratp_955.csv	RATP	0.85	0.92	0.75
1003	dataset_sncf_7702.csv	SNCF	0.60	0.65	0.93

Le script de l'annexe C permet de créer cette table HBase.

4.4.2.2.4. Mise en correspondance de schémas

Cette étape consiste à charger les données et sélectionner les colonnes à évaluer ($X\ \{x_1, \dots, x_n\}$) et faire ensuite correspondre chaque colonne à évaluer à celles existantes dans le Data Lake ($Y\ \{y_1, \dots, y_m\}$). Différents cas peuvent exister :

- Le cas "simple" : à chaque colonne x_i correspond une colonne y_j.
- Le cas "conflictuel" : à chaque colonne x_i corespond un ensemble de colonnes $\{y_j\}$ de cardinalité $k \in [2, p]$ où p est le nombre de sources de données. Pour notre étude de cas, $k \in [2, 3]$. En cas de conflit, c'est la colonne de l'ensemble de données qui répond au mieux aux critères d'exactitude qui sera considérée pour l'opération de mise en correspondance. Pour cela, nous avons mis en place un algorithme de résolution de conflit qui consiste à attribuer un poids à chaque critère d'exactitude. En cas de conflit, c'est l'ensemble de données ayant la somme pondérée la plus élevée des valeurs des critères d'exactitude qui sera considéré pour le processus de mise en correspondance.

- Le cas "incomplet" : il existe un ensemble de colonnes $\{x_i\}$ de cardinalité $l \in [1, q]$, où q est le nombre total des colonnes à évaluer, qui n'a aucune correspondance dans le Data Lake. Notons que plus l est grand (s'approche de q), plus le calcul de l'exactitude perd sa fiabilité. Si $l = q$ le calcul de l'exactitude ne peut être réalisé (la mise en correspondance dans ce cas est impossible et par conséquent on ne peut pas identifier des données de référence ce qui mène à l'échec du processus d'évaluation de l'exactitude).

Voici le schéma de données que l'on souhaite évaluer pour notre étude de cas :

Champ	Descriptif	Format
Nom_Gare	Nom de la gare	String
Commune	la commune de la gare	String
Coordonnees_Geographiques	latitude, longitude	Decimal, Decimal
Annonce_Sonore_Prochain_Passage	Annonce Sonore Prochain Passage	Boolean
Annonce_Visuelle_Prochain_Passage	Annonce Visuelle Prochain Passage	Boolean
Annonce_Sonore_Situations_Perturbees	Annonce Sonore Situations Perturbees	Boolean
Annonce_Visuelle_Situations_Perturbees	Annonce Visuelle Situations Perturbees	Boolean

Les noms des gares et les coordonnées géographiques présentent une situation conflictuelle ; ces colonnes existent dans les trois sources de données. Les autres colonnes ne présentent pas de conflit : les équipements sonores et visuels correspondent à des colonnes de la source de données RATP et la commune correspond à une colonne de la source de données SNCF. Pour notre étude de cas, une table de correspondance serait :

Champ	Île-de-France	RATP	SNCF
Nom_Gare	nom_gare	nomptar	LIBELLE_GARE
Commune	N/A	N/A	COMMUNE
Coordonnees_Geographiques	Geo Point	coord	coordonnees_geographiques
Annonce_Sonore_Prochain_Passage	N/A	AnnonceSonoreProchainPassage	N/A
Annonce_Visuelle_Prochain_Passage	N/A	AnnonceVisuelleProchainPassage	N/A
Annonce_Sonore_Situations_Perturbees	N/A	AnnonceSonoreSituationsPerturbees	N/A
Annonce_Visuelle_Situations_Perturbees	N/A	AnnonceVisuelleSituationsPerturbees	N/A

Sutanta et al. [Sutanta, 16], en référence à d'autres travaux sur les correspondances de schémas, ont réalisé une étude comparative basée sur 34 modèles prototypes. Ces modèles considèrent différents aspects tels que le type de saisie, les algorithmes utilisés, le domaine d'utilisation, etc. Selon cette étude, l'un des prototypes de correspondance de schémas les plus réussis est COMA 3.0 [Rahm, 11], qui est une évolution de COMA++ [Aumueller, 05]. Ce prototype accepte différents types de données d'entrée (XSD, XDR, OWL, CSV, SQL), utilise différents algorithmes de correspondance (basé sur la linguistique, la structure et l'instance), n'est pas spécifique à un domaine d'utilisation particulier et interactif via une interface graphique. C'est donc ce prototype que nous adoptons pour réaliser la mise en correspondance

des schémas de données à évaluer et celles présentes sur le Data Lake. La Figure 4.4 est une capture d'écran de la GUI COMA 3.0 de notre processus de mise en correspondance.

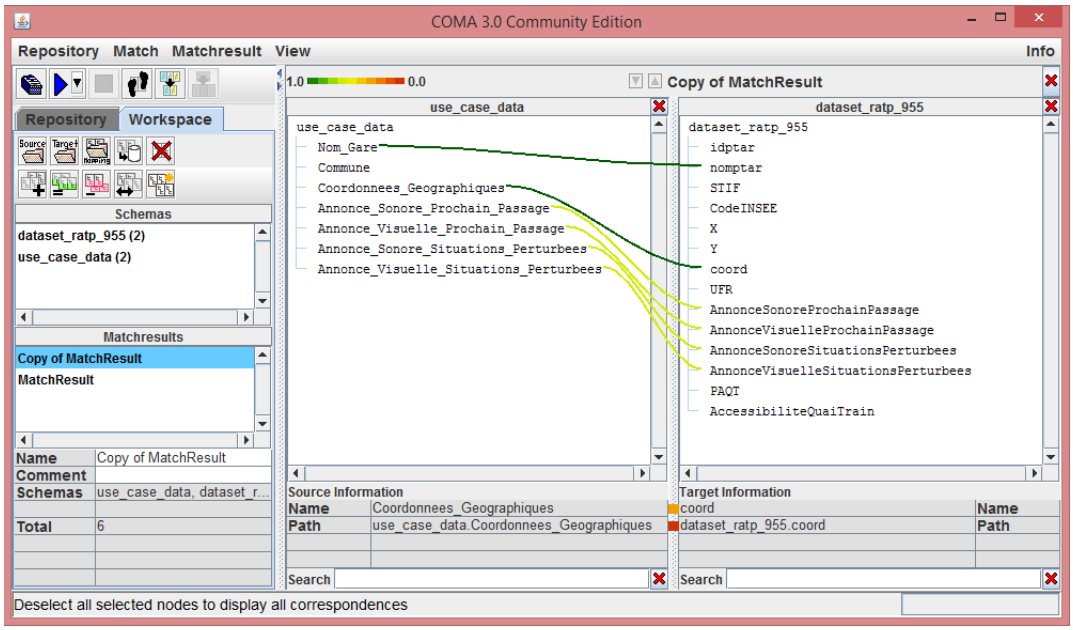

Figure 4.4. GUI COMA 3.0

4.4.2.2.5. Identification des données de référence et Calcul de l'exactitude

Une fois que les critères d'exactitude ont été valorisés et normalisés et que les correspondances entre les colonnes à évaluer et celles du Data Lake ont été établies, nous avons pu procéder à l'identification des données de référence. Avant de procéder au couplage des enregistrements, des opérations de résolution de conflit avaient été nécessaires. Afin de déduire des conclusions de cette étude, nous avons fusionné l'étape d'identification des données de référence et celle de calcul de la valeur de l'exactitude. Nous avons exécuté une procédure expérimentale de sorte que pour chaque itération nous calculons et nous sauvegardons la valeur de l'exactitude. Pour ce faire, pour chaque valeur normalisée du triplet {T, R, C} allant de {0.00, 0.00, 0.00} jusqu'à {1.00, 1.00, 1.00}, avec un pas de 0.05, nous avons réalisé les opérations suivantes :

- Résoudre les éventuels conflits de la table de correspondance. La résolution des conflits consiste à ce que chaque colonne des données à évaluer corresponde à au plus une colonne des données du Data Lake. Si une colonne correspond à plusieurs colonnes, c'est celle appartenant au jeu de données ayant la moyenne pondérée la plus élevée pour les trois critères d'exactitude qui est considérée.
- Appliquer un mécanisme de couplage d'enregistrements entre les données à évaluer et celles enregistrées dans la table de correspondance nettoyée. Nous avons mis en place ce mécanisme grâce à la bibliothèque RecordLinkage[45] du langage python qui fournit des méthodes d'indexation, des fonctions de mesures de similarité et des classificateurs.
- Déduire les données de référence. Il s'agit du résultat du processus de couplage d'enregistrements.

[45] https://pypi.org/project/recordlinkage/

- Calculer et sauvegarder la valeur de l'exactitude en mesurant le degré de similarité entre les enregistrements à évaluer et ceux de référence. Chaque paire d'enregistrements est une correspondance candidate. Pour classer les paires d'enregistrements candidats en "correspondances" et en "non-correspondances", il faut comparer les enregistrements des attributs de la table de correspondance. La bibliothèque RecordLinkage adoptée pour cette étude présente une classe nommée *Comparer* qui permet de comparer les attributs des enregistrements tout en choisissant la méthode adéquate à chaque type de données (chaînes de caractères, nombres, positions géographiques, etc.). En calculant les similarités de toutes les valeurs de tous les enregistrements candidats, nous pouvons déduire l'exactitude pour chaque colonne ainsi que pour toute la table à évaluer.

4.4.2.3. Validation de l'implémentation du concept

Pour valider notre implémentation, nous avons procédé à des tests unitaires comme suit :

- Créer un Data Lake avec un seul ensemble de données (*dataset_ratp_955.csv*). Dans la table de métadonnées HDFS_METADATA, nous avons assigné des valeurs maximales pour chaque critère d'exactitude (Confiance = 100%, Réputation = 5/5, Cohérence = 100%). Le but étant d'expérimenter le calcul de l'exactitude sur un ensemble de données unique dont on connait à priori les résultats.
- A partir du fichier *dataset_ratp_955.csv*, nous avons extrait 100 lignes et l'avons enregistré dans un nouveau fichier *data_ratp_100.csv* qui sera le fichier des données à évaluer. A ce niveau, le fichier *data_ratp_100.csv* est entièrement inclus dans *dataset_ratp_955.csv*.
- Pour notre test de validation, nous nous sommes intéressés à l'étude de l'exactitude de six colonnes uniquement, soient 600 valeurs (100 lignes multipliés par 6 colonnes). Nous avons lancé le calcul de l'exactitude du fichier *data_ratp_100.csv* sans ne lui apporter de modification, le résultat devrait être une exactitude de 100%. Nous avons ensuite mis en erreur un certain nombre de valeurs et relancé le calcul de l'exactitude ; le nouveau résultat devait tenir en considération le nombre d'informations erronées. Nous avons répété cette action plusieurs fois et à chaque reprise nous nous sommes assurés que l'exactitude correspondait bien au nombre de valeurs correctes dans le fichier *data_ratp_100.csv*.

Voici les étapes et les résultats de nos tests unitaires :

- **Test 1** : les 600 valeurs du fichier *data_ratp_100.csv* sont correctes. Le calcul de l'exactitude nous a bien retourné 1 (donc 100% des données sont correctes).
- **Test 2** : mise en erreur de 30 valeurs choisies aléatoirement dans les six colonnes qui nous intéressent dans *data_ratp_100.csv*, soient 5% des données sont erronées. Le calcul de l'exactitude nous a bien retourné 0.95 (donc 95% des données sont correctes).
- **Test 3** : mise en erreur de 50 valeurs choisies aléatoirement dans les six colonnes qui nous intéressent dans *data_ratp_100.csv*, soient environ 8.34% des données sont erronées. Le calcul de l'exactitude nous a bien retourné 0.91666667 (donc 91.66% des données sont correctes).
- **Test 4** : mise en erreur de 118 valeurs choisies aléatoirement dans les six colonnes qui nous intéressent dans *data_ratp_100.csv*, soient environ 19.67% des données sont erronées. Le calcul de l'exactitude nous a bien retourné 0.80333333 (donc 80.33% des données sont correctes).

Les résultats des tests unitaires permettent de valider notre implémentation. Nous pouvons dès à présent présenter les scénarios des tests de la preuve du concept et analyser les résultats pour arriver aux conclusions.

4.4.2.4. Analyse et Discussion

L'objectif de cette section est d'étudier différents scénarios afin de démontrer la relation entre les critères d'exactitude, d'une part, et la valeur de l'exactitude, d'autre part. Lors de la présentation de notre étude de cas, nous avons choisi trois critères d'exactitude : la Confiance **T**, la réputation **R** et la cohérence **C**. Nous avons également souligné que ces critères peuvent changer selon le contexte de chaque projet ; d'autres critères peuvent s'avérer plus pertinents tels que l'actualité et l'exhaustivité de données. Par ailleurs, les métriques pour mesurer chacun de ces critères dépendent également du contexte de chaque projet. L'essentiel à retenir c'est que chaque source de données du Data Lake doit avoir une valeur correctement calculée pour chaque critère d'exactitude. Nous avons présenté quelques techniques efficaces pour mesurer chacun de nos critères d'exactitude. Pour notre étude de cas, nous avons appliqué des algorithmes hypothétiques ; l'objectif étant de mettre en place un outil de calcul de l'exactitude.

Reprenons le périmètre de notre étude :

- **DS1** : un premier ensemble de données représenté par le fichier *dataset_idf_1140.csv*.
- **DS2** : un deuxième ensemble de données représenté par le fichier *dataset_ratp_955.csv*.
- **DS3** : un troisième ensemble de données représenté par le fichier *dataset_sncf_7702.csv*.
- **Data** : les données d'entrée pour lesquelles nous souhaitons évaluer l'exactitude. Pour cela, nous reprenons le fichier du 4ème Test de nos tests de validation. Nous savons à priori que, si la valeur de l'exactitude de **DS2** est égale à 100%, alors la valeur de l'exactitude de **Data** est **80.33%**.

4.4.2.4.1. Présentation des scénarios de l'étude

Afin de comprendre la relation entre les critères d'exactitude et la valeur de l'exactitude calculée, nous devions suivre une logique qui nous a permis de mesurer l'exactitude en fonction de toutes les données qui existent dans le Data Lake. Pour cela, nous avons mis en place différents scénarios qui nous ont permis de réaliser la mise en correspondance avec tous les ensembles de données du Data Lake (DS1, DS2 et DS3) en fonction des valeurs maximales du triplé {T, R, C}. Ayant trois ensembles de données et trois critères d'exactitude, 27 scénarios sont possibles ($3^3 = 27$) que nous représentons dans la Table 4.2.

Table 4.2. Les 27 scénarios de notre étude de cas

	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
Scénario 1	1	1	1	0	0	0	0	0	0
Scénario 2	0	0	0	1	1	1	0	0	0
Scénario 3	0	0	0	0	0	0	1	1	1
Scénario 4	1	1	0	0	0	1	0	0	0
Scénario 5	1	1	0	0	0	0	0	0	1
Scénario 6	0	0	1	1	1	0	0	0	0
Scénario 7	0	0	0	1	1	0	0	0	1
Scénario 8	0	0	1	0	0	0	1	1	0

Scénario 9	0	0	0	0	0	1	1	1	0
Scénario 10	1	0	1	0	1	0	0	0	0
Scénario 11	1	0	1	0	0	0	0	1	0
Scénario 12	0	1	0	1	0	1	0	0	0
Scénario 13	0	0	0	1	0	1	0	1	0
Scénario 14	0	1	0	0	0	0	1	0	1
Scénario 15	0	0	0	0	1	0	1	0	1
Scénario 16	0	1	1	1	0	0	0	0	0
Scénario 17	0	1	1	0	0	0	1	0	0
Scénario 18	1	0	0	0	1	1	0	0	0
Scénario 19	0	0	0	0	1	1	1	0	0
Scénario 20	1	0	0	0	0	0	0	1	1
Scénario 21	0	0	0	1	0	0	0	1	1
Scénario 22	1	0	0	0	1	0	0	0	1
Scénario 23	1	0	0	0	0	1	0	1	0
Scénario 24	0	1	0	1	0	0	0	0	1
Scénario 25	0	1	0	0	0	1	1	0	0
Scénario 26	0	0	1	1	0	0	0	1	0
Scénario 27	0	0	1	0	1	0	1	0	0

Le scénario 1 consiste à ce que les valeurs maximales du triple {T, R, C} soient assignées à l'ensemble de données DS1. Cela garantit que la correspondance soit réalisée avec DS1 et par conséquent toutes les données de référence soient forcément extraites de DS1. De la même façon, les scénarios 2 et 3 garantissent que la correspondance soit réalisée respectivement avec les ensembles de données DS2 et DS3 et par conséquent les données de référence soient extraites de DS2 et DS3 respectivement. Pour les scénarios 4 à 9, nous avons assigné les valeurs maximales de {T, R} à l'un des trois ensembles de données (par exemple DS1 pour les scénarios 4 et 5) et la valeur maximale de {C} à l'un des autres ensembles de données (par exemple DS2 pour le scénario 4 et DS3 pour le scénario 5). Nous avons suivi ensuite la même logique pour les scénarios 10 à 21. Enfin, pour les scénarios 22 à 27, un des trois ensembles de données avait une, et une seule, valeur maximale des trois critères d'exactitude. Cela garantit que les données de référence soient sélectionnées de tous les ensembles de données (DS1, DS2 et DS3).

L'étape suivante consiste à assigner des valeurs hypothétiques à chaque cellule de la table précédente. Nous considérons que :

- La confiance T est représentée par un pourcentage $\in$ [0, 100]
- La réputation R prend une valeur numérique $\in$ [1, 5]
- La cohérence C est représentée par un pourcentage $\in$ [0, 100]

Un cas particulier qui n'est couvert par aucun des scénarios précédents est le cas où aucun ensemble de données ne satisfait tous les critères d'exactitude requis par un cas d'utilisation. Pour pouvoir étudier le comportement pour ce cas particulier, nous avons attribué à chaque critère d'exactitude des valeurs proches, mais inférieures, à la valeur maximale possible. La Table 4.3 représente les valeurs hypothétiques pour chaque critère d'exactitude pour chaque scénario.

Table 4.3. Attribution des valeurs hypothétiques pour chaque critère d'exactitude

	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
Scénario 1	90	4,4	95	35	2	45	30	1,8	40
Scénario 2	40	2,2	45	85	4,2	90	30	1,8	40
Scénario 3	40	2,2	45	35	2	45	80	4	85
Scénario 4	90	4,4	45	35	2	90	30	1,8	40
Scénario 5	90	4,4	45	35	2	45	30	1,8	85
Scénario 6	40	2,2	95	85	4,2	45	30	1,8	40
Scénario 7	40	2,2	45	85	4,2	45	30	1,8	85
Scénario 8	40	2,2	95	35	2	45	80	4	40
Scénario 9	40	2,2	45	35	2	90	80	4	40
Scénario 10	90	2,2	95	35	4,2	45	30	1,8	40
Scénario 11	90	2,2	95	35	2	45	30	4	40
Scénario 12	40	4,4	45	85	2	90	30	1,8	40
Scénario 13	40	2,2	45	85	2	90	30	4	40
Scénario 14	40	4,4	45	35	2	45	80	1,8	85
Scénario 15	40	2,2	45	35	4,2	45	80	1,8	85
Scénario 16	40	4,4	95	85	2	45	30	1,8	40
Scénario 17	40	4,4	95	35	2	45	80	1,8	40
Scénario 18	90	2,2	45	35	4,2	90	30	1,8	40
Scénario 19	40	2,2	45	35	4,2	90	80	1,8	40
Scénario 20	90	2,2	45	35	2	45	30	4	85
Scénario 21	40	2,2	45	85	2	45	30	4	85
Scénario 22	90	2,2	45	35	4,2	45	30	1,8	85
Scénario 23	90	2,2	45	35	2	90	30	4	40
Scénario 24	40	4,4	45	85	2	45	30	1,8	85
Scénario 25	40	4,4	45	35	2	90	80	1,8	40
Scénario 26	40	2,2	95	85	2	45	30	4	40
Scénario 27	40	2,2	95	35	4,2	45	80	1,8	40

Pour faire abstraction aux domaines de définition de chaque critère, nous avons normalisé les valeurs (en appliquant la technique Min-Max Scaling). La Table 4.4 représente les valeurs normalisées pour chaque critère d'exactitude (les valeurs maximales pour chaque scénario sont colorées en vert).

Table 4.4. Normalisation des valeurs hypothétiques de chaque critère d'exactitude

	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
Scénario 1	0.9	0.85	0.95	0.35	0.25	0.45	0.3	0.2	0.4
Scénario 2	0.4	0.3	0.45	0.85	0.8	0.9	0.3	0.2	0.4
Scénario 3	0.4	0.3	0.45	0.35	0.25	0.45	0.8	0.75	0.85
Scénario 4	0.9	0.85	0.45	0.35	0.25	0.9	0.3	0.2	0.4

Scénario 5	0.9	0.85	0.45	0.35	0.25	0.45	0.3	0.2	0.85
Scénario 6	0.4	0.3	0.95	0.85	0.8	0.45	0.3	0.2	0.4
Scénario 7	0.4	0.3	0.45	0.85	0.8	0.45	0.3	0.2	0.85
Scénario 8	0.4	0.3	0.95	0.35	0.25	0.45	0.8	0.75	0.4
Scénario 9	0.4	0.3	0.45	0.35	0.25	0.9	0.8	0.75	0.4
Scénario 10	0.9	0.3	0.95	0.35	0.8	0.45	0.3	0.2	0.4
Scénario 11	0.9	0.3	0.95	0.35	0.25	0.45	0.3	0.75	0.4
Scénario 12	0.4	0.85	0.45	0.85	0.25	0.9	0.3	0.2	0.4
Scénario 13	0.4	0.3	0.45	0.85	0.25	0.9	0.3	0.75	0.4
Scénario 14	0.4	0.85	0.45	0.35	0.25	0.45	0.8	0.2	0.85
Scénario 15	0.4	0.3	0.45	0.35	0.8	0.45	0.8	0.2	0.85
Scénario 16	0.4	0.85	0.95	0.85	0.25	0.45	0.3	0.2	0.4
Scénario 17	0.4	0.85	0.95	0.35	0.25	0.45	0.8	0.2	0.4
Scénario 18	0.9	0.3	0.45	0.35	0.8	0.9	0.3	0.2	0.4
Scénario 19	0.4	0.3	0.45	0.35	0.8	0.9	0.8	0.2	0.4
Scénario 20	0.9	0.3	0.45	0.35	0.25	0.45	0.3	0.75	0.85
Scénario 21	0.4	0.3	0.45	0.85	0.25	0.45	0.3	0.75	0.85
Scénario 22	0.9	0.3	0.45	0.35	0.8	0.45	0.3	0.2	0.85
Scénario 23	0.9	0.3	0.45	0.35	0.25	0.9	0.3	0.75	0.4
Scénario 24	0.4	0.85	0.45	0.85	0.25	0.45	0.3	0.2	0.85
Scénario 25	0.4	0.85	0.45	0.35	0.25	0.9	0.8	0.2	0.4
Scénario 26	0.4	0.3	0.95	0.85	0.25	0.45	0.3	0.75	0.4
Scénario 27	0.4	0.3	0.95	0.35	0.8	0.45	0.8	0.2	0.4

L'exécution des 27 scénarios a duré 34435 secondes (soient 09 heures, 33 minutes et 55 secondes). Pour chaque scénario, nous avions 9261 itérations (pour chaque variable T, R et C, de 0.00 à 1.00 avec un pas de 0.05, nous avions 21 itérations ce qui fait un total de 21^3 = 9261 itérations). Nous avons donc adapté notre programme pour boucler sur cette table, calculer et éditer la valeur de l'exactitude pour chaque itération de chaque scénario.

4.4.2.4.2. Présentation des résultats

Suite à notre analyse des résultats, nous avons remarqué que les scénarios 1 à 3 peuvent être regroupés dans un même groupe A puisqu'ils donnent le même comportement, les scénarios 4 à 21 peuvent être regroupés dans un même groupe B et les scénarios 22 à 27 dans le groupe C. Ainsi, dans cette section, nous nous contentons de présenter les résultats de trois scénarios : le scénario 1 (Figure 4.5, Table 4.5 et Table 4.6), le scénario 4 (Figure 4.6, Table 4.7 et Table 4.8) et le scénario 22 (Figure 4.7, Table 4.9 et Table 4.10). Tous les autres scénarios sont présentés dans l'annexe D.

Nous présentons les résultats sur un repère à quatre dimensions (Trust, Reputation, Consistency et Accuracy). Pour chaque scénario, nous présentons le temps de calcul, les valeurs de chaque critère d'exactitude pour chaque ensemble de données ainsi que leur somme pondérée qui permet de résoudre les éventuels conflits (pour simplifier nos analyses, nous avons supposé que le poids est égal à 1 pour les trois critères d'exactitude). Pour chaque scénario, on trouve plusieurs valeurs possibles selon des conditions

représentées par des plages de valeurs pour chaque critère d'exactitude. Si la valeur de l'exactitude vaut NONE c'est qu'aucun ensemble de données ne répond aux valeurs minimales requises pour les critères d'exactitude.

4.4.2.4.2.1. Scénario 1

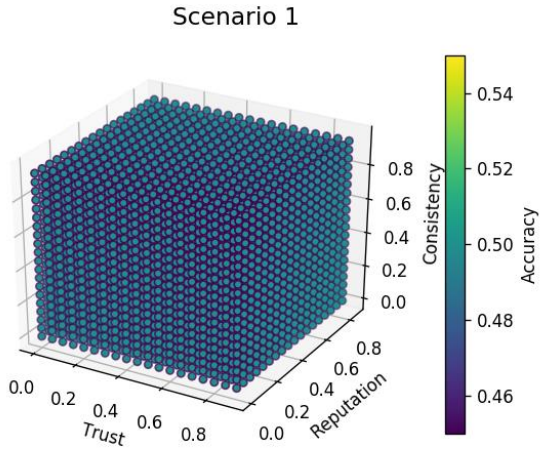

Figure 4.5. Résultats de calcul de l'exactitude de données pour le Scénario 1

Table 4.5. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 1

Temps d'exécution : 1963 Seconds [32 Minutes, 43 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.9	0.85	0.95	0.35	0.25	0.45	0.3	0.2	0.4
Somme pondérée	2.70			1.05			0.90		

Table 4.6. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 1

Exactitude	Occurrences	Conditions
None	2421	T > 0.90 || R > 0.85 || C > 0.95
0.50	6840	T ≤ 0.90 && R ≤ 0.85 && C ≤ 0.95

4.4.2.4.2.2. Scénario 4

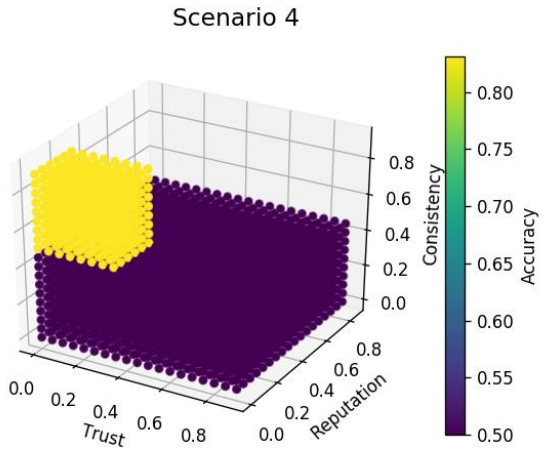

Figure 4.6. Résultats de calcul de l'exactitude de données pour le Scénario 4

Table 4.7. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 4

Temps d'exécution : 1266 Seconds [21 Minutes, 06 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.9	0.85	0.45	0.35	0.25	0.9	0.3	0.2	0.4
Somme pondérée	2.20			1.50			0.90		

Table 4.8. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 4

Exactitude	Occurrences	Conditions
None	5409	T > 0.90 || R > 0.85 || C > 0.90 || (C > 0.45 && R > 0.25) || (C > 0.45 && T > 0.35)
0.50	3420	T ≤ 0.90 && R ≤ 0.85 && C ≤ 0.45
0.8314	432	T ≤ 0.35 && R ≤ 0.25 && 0.45 < C ≤ 0.90

4.4.2.4.2.3. Scénario 22

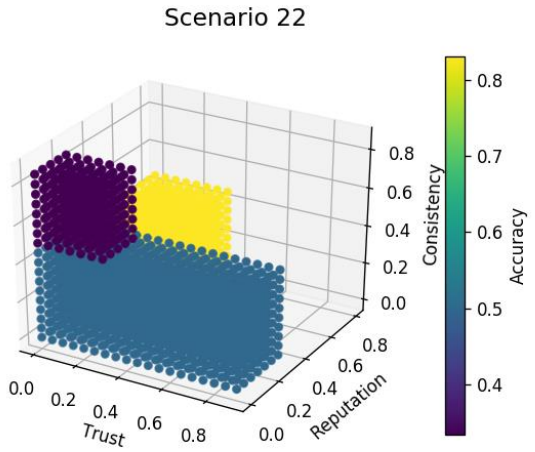

Figure 4.7. Résultats de calcul de l'exactitude de données pour le Scénario 22

Table 4.9. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 22

Temps d'exécution : 986 Seconds [16 Minutes, 26 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.9	0.3	0.45	0.35	0.8	0.45	0.3	0.2	0.85
Somme pondérée	1.65			1.60			1.35		

Table 4.10. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 22

Exactitude	Occurrences	Conditions
None	6851	T > 0.90 || R > 0.80 || C > 0.85 || (C > 0.45 && R > 0.20) || (C > 0.45 && T > 0.30)
0.50	1330	T ≤ 0.90 && R ≤ 0.30 && C ≤ 0.45
0.8314	800	T ≤ 0.35 && 0.30 < R ≤ 0.80 && C ≤ 0.45
0.3333	280	T ≤ 0.30 && R ≤ 0.20 && 0.45 < C ≤ 0.85

4.4.2.4.3. Analyse des résultats

La clé de notre solution réside dans les données de référence. L'exactitude est mesurée en calculant la similarité des données à évaluer avec celles de référence. Pour sélectionner ces données de référence, une étape de mise en correspondance est réalisée afin de faire correspondre les colonnes des données à évaluer avec celles des ensembles de données du Data Lake. Le processus de mise en correspondance peut engendrer des conflits. C'est le cas où une colonne des données à évaluer est liée à plusieurs colonnes dans différents ensembles de données du Data Lake. Un tel conflit empêche la sélection des données de référence et requiert sa résolution. Pour notre étude de cas, nous proposons d'assigner à chaque critère d'exactitude un poids représentant son importance dans le processus global d'évaluation de l'exactitude de données et de comparer ensuite la somme pondérée des critères d'exactitude. En cas de conflit, le jeu de données ayant la plus grande somme pondérée sera considéré dans le processus de sélection des données de référence. Ce mécanisme n'est qu'une proposition qui nous permet d'atteindre notre objectif ; les règles de résolution du conflit dépendent du contexte de chaque projet. En analysant toutes les itérations des scénarios de notre étude, nous constatons que l'exactitude peut prendre une des quatre valeurs suivantes :

- 50% : quand les données de référence sont extraites de DS1.
- 83.14% : quand les données de référence sont extraites de DS2.
- 33.33% : quand les données de référence sont extraites de DS3.
- None : si aucun jeu de données ne répond aux critères requis par une itération, la valeur de l'exactitude ne peut être calculée. Notre solution retourne alors la valeur « None ».

En outre, nous pouvons classer les 27 scénarios en 3 groupes :

- **Groupe A (Scénario 1 à 3)** : pour ces scénarios, un, et un seul, jeu de données déteint toutes les valeurs maximales des critères d'exactitude. La Table 4.11 résume les résultats de ce groupe.

Table 4.11. Nombre d'itérations pour chaque valeur d'exactitude pour les scénarios du Groupe A

Group A	None	50%	83,14%	33,33%
Scénario 1	2421	6840	0	0
Scénario 2	3447	0	5814	0
Scénario 3	4365	0	0	4896

A partir de cette table, nous avons construit le diagramme radar illustré dans la Figure 4.8.

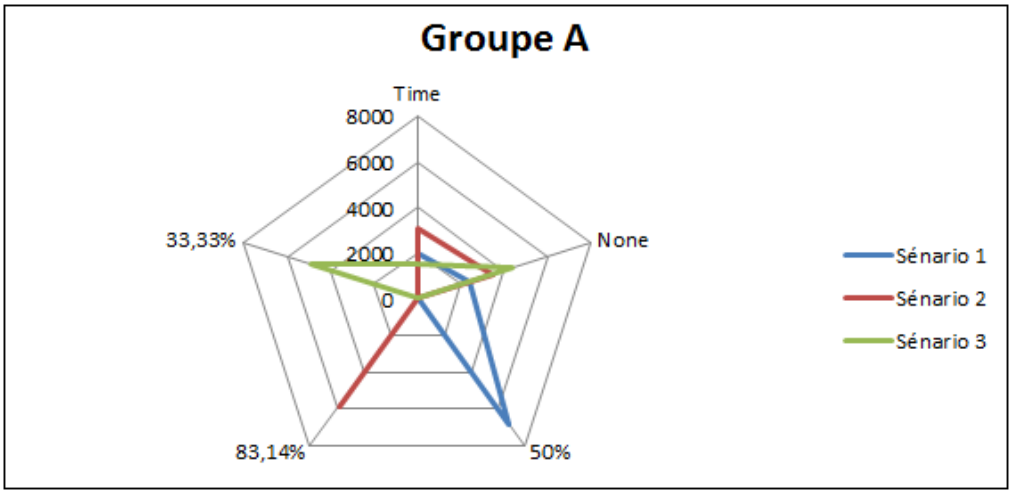

Figure 4.8. Représentation des résultats du Groupe A

- **Groupe B (Scénario 4 à 21)** : pour ces scénarios, les données de référence sont extraites de deux jeux de données. La Table 4.12 résume les résultats de ce groupe.

Table 4.12. Nombre d'itérations pour chaque valeur d'exactitude pour les scénarios du Groupe B

Groupe B	None	50%	83,14%	33,33%
Scénario 4	5409	3420	432	0
Scénario 5	5561	3420	0	280
Scénario 6	5571	630	3060	0
Scénario 7	5921	0	3060	280
Scénario 8	6120	693	0	2448
Scénario 9	6318	15	480	2448
Scénario 10	5801	2660	800	0
Scénario 11	6034	2660	0	567
Scénario 12	6129	1080	2052	0
Scénario 13	6552	27	2052	630
Scénario 14	6561	1170	0	1530
Scénario 15	6751	20	960	1530
Scénario 16	5481	3240	540	0
Scénario 17	5661	3240	0	360
Scénario 18	5907	770	2584	0
Scénario 19	6247	25	2584	405
Scénario 20	6405	840	0	2016
Scénario 21	6565	20	660	2016

A partir de cette table, nous avons construit le diagramme radar illustré dans la Figure 4.9.

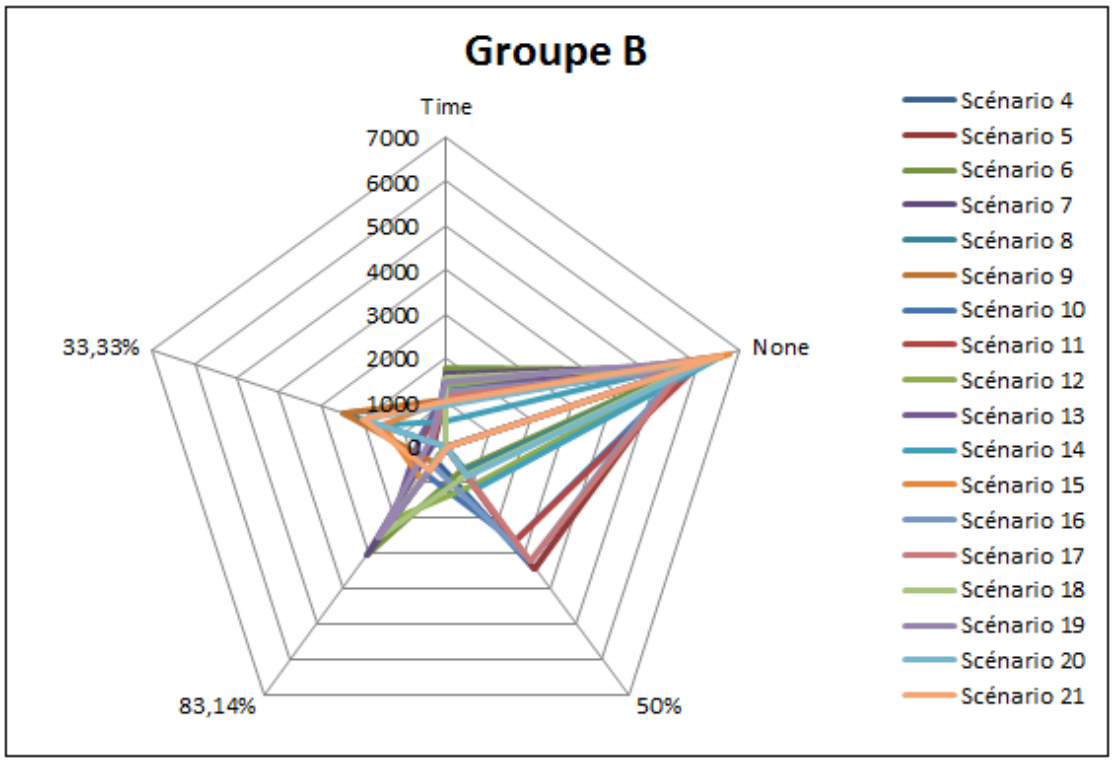

Figure 4.9. Représentation des résultats du Groupe B

- **Groupe C (Scénario 22 à 27)** : pour ces scénarios, un jeu de données ne peut avoir qu'une seule des valeurs maximales des critères d'exactitude. Ceci dit, les données de référence sont sélectionnées des trois ensembles de données. Cela implique une grande hétérogénéité dans les données de référence et, par conséquent, différentes valeurs unitaires pour l'exactitude. La Table 4.13 résume les résultats de ce groupe.

Table 4.13. Nombre d'itérations pour chaque valeur d'exactitude pour les scénarios du Groupe C

Groupe C	None	50%	83,14%	33,33%
Scénario 22	6851	1330	800	280
Scénario 23	6932	1330	432	567
Scénario 24	6821	1620	540	280
Scénario 25	6849	1620	432	360
Scénario 26	6894	1260	540	567
Scénario 27	6841	1260	800	360

A partir de cette table, nous avons construit le diagramme radar illustré dans la Figure 4.10.

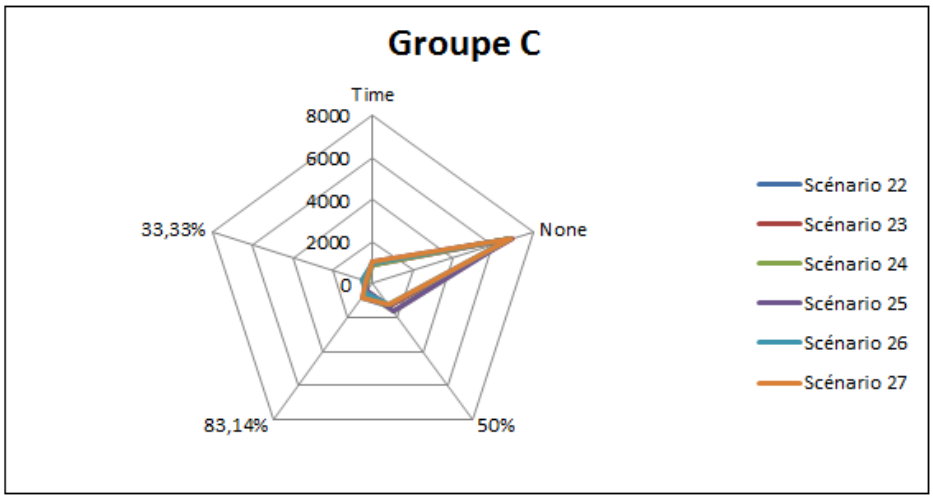

Figure 4.10. Représentation des résultats du Groupe C

L'analyse de ces trois diagrammes radar nous permet de déduire les constats suivants :

- Plus les valeurs maximales des critères d'exactitude sont détenues par un minimum de jeux de données, plus l'exactitude de données à évaluer est calculée en fonction de ces jeux de données. Cela minimise le nombre des valeurs « None » ce qui améliore la fiabilité du système d'évaluation. En revanche, le temps est plus élevé. Ceci est expliqué par le grand nombre de données pour lequel il faut appliquer le couplage d'enregistrements, la sélection et l'analyse des données de référence.
- Plus les valeurs maximales des critères d'exactitude sont éparpillées sur plusieurs jeux de données, plus le calcul de l'exactitude est complexe. Cela minimise la fiabilité de la solution mais permet d'avoir des résultats plus rapidement.
- La résolution des conflits de correspondance est étroitement liée aux considérations de chaque domaine d'application de la solution.
- Le choix et les mesures des bons critères d'exactitude impactent directement la fiabilité et la robustesse de la solution.

- La définition des seuils minimums pour chaque critère d'exactitude à partir desquels une source de données est considérée comme fiable, agit sur le choix des données de référence et, par conséquent, la possibilité de calcul de l'exactitude.

4.5. Conclusion

Suite à cette étude, nous concluons que la mise en place d'un système d'évaluation de l'exactitude de données dans un environnement Big Data dépend de plusieurs éléments relatifs principalement au contexte de chaque projet. Voici les principales étapes pour mettre en place un tel système :

- Avoir un Data Lake "riche" en données de bonne qualité.
- Comprendre les schémas de tous les ensembles de données composant le Data Lake.
- Définir les critères d'exactitude qui caractérisent au mieux les données de bonne qualité.
- Déterminer les seuils minimums pour chaque critère d'exactitude pour considérer une source de données comme fiable.
- Définir les règles de résolution des conflits de correspondance.
- Implémenter la solution dans un environnement distribué.

L'évaluation de la qualité de données peut nécessiter un accès en lecture à l'ensemble des données hébergées dans le lac de données d'une organisation. L'amélioration de la qualité de données, quant à elle, requiert un accès en écriture aux données. En revanche, comme nous l'avons détaillé dans [Talha, 19], la confidentialité des données implique la mise en place d'un ensemble de techniques, de règles et de restrictions pour limiter l'accès aux données confidentielles. Dans un contexte Big Data, la mise en place des mécanismes de confidentialité doit relever un ensemble de défis tels que la fusion et l'intégration des stratégies de contrôles d'accès, la gestion automatique de ces stratégies, l'application du contrôle d'accès sur des plateformes Big Data, etc. [Bertino, 15]. Tous ces défis de sécurité peuvent rendre les processus de gestion de la qualité de données plus lents et plus complexes.

Pour notre solution d'évaluation de la qualité de données, nous sommes partis du principe que l'ensemble des données du Data Lake sont accessibles en lecture ce qui n'est généralement pas le cas dans un cadre professionnel. En effet, la mise en place d'une infrastructure Big Data nécessite des budgets importants. Une des solutions les plus pratiquées aujourd'hui pour baisser les coûts d'investissement consiste à cohabiter les données de plusieurs projets et applications au sein d'un même Data Lake. Les données se retrouvent alors protégées par des mécanismes et des politiques de sécurité différents même si elles appartiennent à la même organisation. Dans la suite de nos travaux, nous allons nous concentrer sur les systèmes de sécurité de données en Big Data dans le but de trouver un rapprochement avec les systèmes de gestion de la qualité de données.

Chapitre 5. Contrôle d'accès dynamique pour les environnements collaboratifs

5.1. Introduction

Dans un environnement collaboratif, un acteur peut être fournisseur ou consommateur de données vis-à-vis des autres acteurs. Cette relation fournisseur/consommateur est régie par un contrat (ou accord) d'accès composé d'un ensemble d'exigences fonctionnelles et non-fonctionnelles cadrant les modalités d'accès aux données entre les organisations. Par exigence fonctionnelle, on désigne tous les aspects techniques et non-techniques (format de données, protocole d'accès, mécanismes d'authentifications, URL, URI, etc.) qui permettent d'assurer l'accès aux données. Par exigence non-fonctionnelle, on désigne les engagements du fournisseur en termes de qualité de services (disponibilité, latence, sécurité, etc.) ainsi que les métriques et les moyens pour surveiller les engagements, les clauses de recours aux sanctions, le paiement, etc. Tous ces aspects représentent les termes d'un contrat d'accès que l'on doit créer, négocier et surveiller. A l'ère de la Big Data, de l'IoT et du Cloud Computing, les architectures sont de plus en plus dynamiques et distribuées ce qui impose en conséquence l'automatisation de la gestion des contrats d'accès entre les organisations collaboratives. Cela appelle à réfléchir à des systèmes de contrôle d'accès dynamiques pour servir ces types d'environnements. Les accords d'accès, en revanche, doivent être gérés automatiquement, selon un concept Machine-to-Machine, tout au long de leur cycle de vie (de la négociation jusqu'à la surveillance).

L'objectif de ce chapitre est de démontrer le but et les moyens d'étendre et de combiner deux solutions existantes : le cadre PolyOrBAC [Abou El Kalam, 07] [Abou El Kalam, 09] et la spécification WS-Agreement [Andrieux, 04] au travers de son implémentation SLA-Framework [SLA-Framework, 16]. PolyOrBAC est destiné à contrôler les accès dans les systèmes collaboratifs via des services web. Il permet d'implémenter des politiques de contrôle d'accès locales, pour toute organisation dans un environnement collaboratif, ainsi que des règles de collaboration impliquant plusieurs organisations. PolyOrBAC souffre d'une limitation majeure : la négociation et la création des accords d'accès est un processus manuel mené en amont par les acteurs impliqués dans l'activité collaborative. Quant à la spécification WS-Agreement, elle est actuellement la norme en termes de contrats établis entre les fournisseurs et les consommateurs des services web. Cette spécification définit un protocole et un langage pour la négociation dynamique, la renégociation, la création et la surveillance des accords d'accès dans les systèmes distribués. Dans [Talha, 21-a], nous avons proposé une extension et une combinaison des cadres PolyOrBAC et SLA-Framework. Le but étant d'automatiser le mécanisme de gestion des accords d'accès tout au long de leur cycle de vie et de contrôler ainsi, de façon dynamique, les accès dans les plateformes collaboratives.

Le reste de ce chapitre est organisé comme suit. La section 2 passe en revue certains travaux similaires, notamment ceux étendant le modèle OrBAC [Abou El Kalam, 03-b] et le cadre PolyOrBAC pour rendre dynamique les environnements collaboratifs ainsi que ceux utilisant la spécification WS-Agreement pour intégrer les aspects de sécurité dans les contrats SLA (Service Level Agreement). La section 3 présente le modèle OrBAC et le cadre PolyOrBAC. La section 4 présente la spécification WS-Agreement et deux de ses implémentations les plus répandues : SLA-Framework et WSAG4J [wsag4j, 12]. La section 5, dédiée à la présentation de l'extension du cadre PolyOrBAC, détaille et valide, à travers une étude de cas, le nouveau protocole de négociation d'accès automatisée de PolyOrBAC. La section 6 montre comment nous avons étendu SLA-Framework pour pouvoir couvrir les exigences de PolyOrBAC. La section 7 est dédiée à la présentation et à l'implémentation de notre cadre de contrôle d'accès dynamique pour les environnements collaboratifs. Enfin, nous concluons ce chapitre dans la section 8.

5.2. Travaux similaires

Les environnements collaboratifs sont caractérisés par l'implication de plusieurs organisations dans une activité pour réaliser un objectif commun. Cette activité collaborative est généralement régie par un accord entre des partenaires sur un ensemble de contraintes telles que les ressources à partager, la durée du contrat, le recours aux sanctions, les règles de sécurité, etc. Les organisations impliquées dans ce type d'activité sont souvent indépendantes les unes des autres et les utilisateurs, inconnus à l'avance par le système, peuvent à tout moment avoir des comportements malicieux (au début ou durant l'activité collaborative). En outre, la concurrence qui peut exister entre ces organisations peut inciter quelques-unes à mener des actions malveillantes. Par conséquent, pour protéger les plateformes collaboratives, le partage des ressources doit se baser sur des règles de restriction formant une politique de sécurité dédiée.

Pour atteindre cet objectif, parmi les solutions les plus discutées dans la littérature on trouve l'intégration de "la gestion de la confiance" entre les entités (organisations ou utilisateurs) impliquées dans une activité collaborative. Généralement, l'établissement d'un système de gestion de confiance consiste à calculer un "score de confiance" tout en se basant sur un ensemble de critères (ou paramètres) mesurables. La confiance peut être évaluée différemment selon le contexte de recherche et les critères de confiance appliqués. Elle peut être une valeur binaire (pour préciser si une entité est digne de confiance ou pas), des valeurs multiples (ex. "très faible", "faible", "moyen", "élevé") ou une valeur continue entre 0 et 1. La confiance peut être appliquée dans différents sens. Dans certains cas, c'est le consommateur qui doit s'assurer que le fournisseur de services possède un niveau de confiance garantissant la qualité et la sécurité des données. Pour cela, le calcul du score de confiance se base sur des critères tels que la fiabilité, la disponibilité, l'intégrité, la maintenabilité [Sun, 11], les recommandations des autres consommateurs de données [Habib, 11] ainsi que le respect des clauses de qualité de service [Saleh, 14]. Dans d'autres cas, c'est plutôt le fournisseur de données qui met en place un système de gestion de confiance pour sécuriser les accès externes. C'est le cas des environnements multi-organisations qui est similaire à notre problématique. Les critères de confiance les plus utilisés pour se protéger contre les actions malveillantes sont "la réputation" des entités externes et "les recommandations" des autres nœuds du système collaboratif. La réputation est le critère le plus utilisé ; elle représente le degré de respect des règles de sécurité imposées par les fournisseurs de données. Elle est généralement mesurée en fonction du comportement et l'historique des transactions des utilisateurs. Certains travaux introduisent "la satisfaction" comme étant un critère de confiance calculé en fonction des deux autres critères, à savoir la réputation et la recommandation.

Afin d'implémenter un système de gestion de confiance, de nombreuses approches réutilisent ou étendent le modèle OrBAC ou le cadre PolyOrBAC. Le modèle OrBAC donne la possibilité de définir des autorisations, des obligations, des interdictions et des recommandations selon le contexte d'accès. Il permet d'appliquer une séparation claire entre les entités concrètes (sujet, action et objet) et les entités abstraites (rôle, activité et vue). Cela donne la possibilité aux organisations appartenant à un même système d'implémenter leurs politiques de sécurité les unes indépendamment des autres. Toutefois, OrBAC présente une limite vis-à-vis des systèmes collaboratifs ; il ne gère pas les accès externes. Cette limite a été résolue grâce au cadre PolyOrBAC qui, grâce à des appels de services web et des agréments d'accès pré-négociés entre les fournisseurs et les consommateurs des services web, permet à un sujet d'accéder aux données d'une organisation à laquelle il n'appartient pas. Cependant, OrBAC et PolyOrBAC ne gèrent pas la confiance entre les différentes entités d'une plateforme collaborative. Dans

[Ait Aali, 15], les auteurs proposent le modèle Trust-PolyOrBAC qui intègre une autorité de certification et d'authentification CAA (pour Certificate Authority Authorization) auprès de laquelle toute entité souhaitant accéder aux services des autres entités doit s'authentifier pour obtenir un identifiant qu'elle utilisera dans ses interactions avec les autres entités. La CAA certifie chaque nouvelle organisation dans le système collaboratif et enregistre tout type de transactions et de configuration d'historique entre les organisations. Toute interaction entre deux organisations est basée sur un score de confiance qui doit dépasser un seuil prédéterminé par la CAA. Ce score de confiance est calculé à base d'un ensemble de paramètres, à savoir la satisfaction des organisations après chaque transaction et la réputation des organisations sollicitées pour leurs recommandations. Dans [Belbergui, 16], les auteurs étendent le modèle OrBAC en introduisant un tiers de confiance TTP (pour Trusted Third Party) pour contrôler les interactions entre les demandeurs d'accès et les plateformes Cloud hébergeant les ressources demandées. Dans leur approche, les auteurs s'inspirent du système français pour la gestion des points de permis de conduire ; chaque nouvel utilisateur se voit attribué un crédit initial de 10 points par le TTP et tous ses comportements sont enregistrés dans un journal d'audit. L'utilisateur est sanctionné pour chaque comportement malveillant et perd des points en fonction de la gravité de violation de la politique de sécurité. En revanche, si son comportement reste correct pendant un certain temps et après un certain nombre d'utilisation du système, il sera récompensé et récupèrera quelques points. Au niveau du calcul de la décision d'accès dans OrBAC, deux utilisateurs ayant le même rôle n'ont pas forcément les mêmes droits d'accès ; les règles de sécurité sont activées ou désactivées en fonction des paramètres de confiance. Sur ce même principe, les modèles TOrBAC [Ben Saidi, 12] et Multi-Trust_OrBAC [Ben Saidi, 13] intègrent un indice de confiance aux règles du modèle OrBAC. L'approche consiste à ce qu'un utilisateur se connecte à un TTP pour créer une session et obtenir un indice de confiance à utiliser pour chaque demande d'accès. L'indice de confiance est initialisé par le gestionnaire de confiance et assigné à chaque nouvel utilisateur. Ensuite, après chaque tentative de violation de la politique de sécurité par un utilisateur, son indice de confiance sera décrémenté par le TTP. Enfin, le modèle Trust-OrBAC [Toumi, 12] évalue la confiance grâce à la notion de "situation" qui correspond au souhait de réaliser une activité sur une vue, et la notion de "temps" représentée par un intervalle de temps pendant lequel l'évaluation de la confiance pour une entité ne change pas. Le modèle Trust-OrBAC étend OrBAC en assignant, par l'administrateur du système, une classe de confiance à chaque rôle. Les utilisateurs et les organisations appartiennent également à des classes de confiance calculée en fonction de l'historique de leurs comportements. Cela permet, lors de l'évaluation d'une requête d'accès, de réaliser "une correspondance" permettant de calculer les rôles qui peuvent être assignés au demandeur d'accès.

Tous les travaux discutés jusqu'ici tentent de gérer la sécurité des activités collaboratives en intégrant la notion de "confiance" par extension du modèle OrBAC ou du cadre PolyOrBAC. Selon nous, du moment où l'expression d'une règle de sécurité dans OrBAC est basée sur une, et une seule organisation, il ne sera pas possible d'adopter OrBAC pour gérer les accès d'une organisation vers une autre. C'est le cas par exemple de Ben Saidi et al. qui ont, dans un premier temps, proposé le modèle TOrBAC [Ben Saidi, 12] pour gérer l'indice de confiance et l'ont ensuite étendu dans Multi-Trust_OrBAC [Ben Saidi, 13] pour couvrir les environnements multi-organisations. En revanche, les travaux étendant le cadre PolyOrBAC arrivent à instaurer, à un certain niveau, une approche dynamique pour gérer la sécurité des données à travers le concept de "confiance". Cela reste toutefois insuffisant pour les environnements collaboratifs tels que les grilles informatiques (Grid Computing), les infrastructures en tant que services et encore plus pour les environnements intelligents. Si on prend PolyOrBAC comme exemple, les activités collaboratives sont régies par des accords "statiques" négociés et conclus au

préalable entre les acteurs. Ces organisations doivent donc avoir un contact direct avant que l'activité collaborative n'ait lieu. Cependant, dans un environnement intelligent, avec un grand nombre d'objets connectés et inconnus les uns des autres, cette approche statique ne peut être adoptée. La résolution de cette problématique consiste à (*i*) automatiser la négociation des règles de sécurité et (*ii*) intégrer les règles de sécurité dans des clauses spécifiques des accords SLA. En accord avec ce principe, Stankov et al. [Stankov, 12] affirment que les SLA peuvent être considérés comme un instrument pour renforcer la confiance entre des fournisseurs et des consommateurs de services dans le Cloud Computing, en particulier au stade initial de l'activité de collaboration, avant qu'une relation ne soit formée.

Une des solutions les plus prometteuses pour réaliser cet objectif est la spécification WS-Agreement. Ludwig et al. [Ludwig, 06] étaient les premiers à exploiter la spécification WS-Agreement pour négocier des contrats SLA. Ils ont ainsi ouvert la voie à l'utilisation des ressources distribuées dans des environnements partagés tels que les infrastructures en tant que service et les grilles informatiques. WS-Agreement offre un mécanisme fiable pour créer des accords électroniques solides entre différentes entités intéressées par la mise en place de collaborations dynamiques qui incluent des obligations mutuelles permettant ainsi de combler le vide existant en essayant d'établir une relation de confiance entre ces entités [Ludwig, 06]. Dans ce contexte, les auteurs de [Smith, 07] présentent une approche, basée sur la spécification WS-Agreement, permettant la mise en place d'un mécanisme de configuration de sécurité à granularité fine en fonction des exigences fournies par les utilisateurs. L'utilisation de la spécification WS-Agreement permet de négocier des paramètres de services traditionnels tels que la configuration des ressources et la qualité de service, ainsi que des paramètres de sécurité tels que le niveau de cryptage et d'isolation des composants du système (sandboxing). Lorsqu'un client souhaite soumettre un travail de calcul ou utiliser un service, il peut créer une instance à partir d'un modèle WS-Agreement pour spécifier ses besoins précis en matière de sécurité et de performances. La spécification WS-Agreement peut également être conjuguée avec les modèles de contrôles d'accès à base d'attributs pour permettre aux fournisseurs et aux consommateurs de services de spécifier leurs règles de sécurité dans les deux sens (puisque les engagements sont mutuels). Li et al. dans [Li, 16] proposent d'intégrer les attributs et leurs valeurs dans un contrat SLA à travers des règles OrBAC. Leur approche consiste à ce que, grâce à la spécification WS-Agreement, les fournisseurs et les consommateurs de services échangent des offres et des contre-offres jusqu'à parvenir à un accord concernant le niveau de service ainsi que les exigences de sécurité liées à l'infrastructure. Ainsi, lors de l'application de l'allocation des ressources, le broker, qui est l'intermédiaire entre le fournisseur et le consommateur des services, prend en considération non seulement les exigences classiques que l'on trouve dans un contrat SLA (qualité de service, garanties, …), mais aussi les règles de sécurité permettant de protéger les différents composants de l'infrastructure. Le choix de WS-Agreement est justifié par son format ouvert permettant d'intégrer différents paramètres, en l'occurrence ceux concernant la sécurité des services.

Toutefois, ces approches n'implémentent pas explicitement la spécification WS-Agreement pour éliminer toute intervention humaine lors de la découverte des services et de la négociation des clauses de l'accord, en particulier ceux relatifs à la sécurité. Elles n'implémentent pas non plus les mécanismes de validation et de signature des contrats SLA. La solution que nous proposons dans ce chapitre permet de répondre à toutes ces exigences et peut être implémentée dans un environnement intelligent dans lequel des objets connectés peuvent conclure et surveiller des agréments d'accès.

5.3. Contrôle d'accès basé sur les organisations

5.3.1. OrBAC (Organization Based Access Control)

OrBAC [Abou El Kalam, 03-b] est un modèle de contrôle d'accès offrant la possibilité d'exprimer des règles contextuelles relatives aux permissions, aux interdictions, aux obligations et aux recommandations en s'appuyant sur un langage formel basé sur la logique du premier ordre. Comme illustré dans la Figure 5.1, l'élément central du modèle OrBAC est l'organisation. Chaque organisation peut être structurée en sous organisations (arborescence) et chacune possède sa propre politique de sécurité.

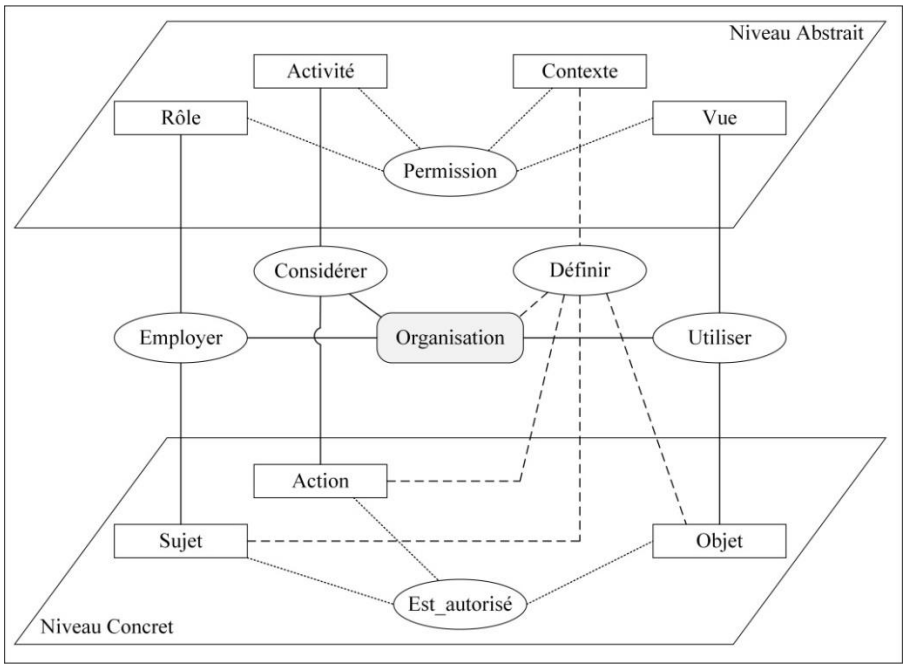

Figure 5.1. Le modèle OrBAC (traduit de [Abou El Kalam, 07])

Le modèle OrBAC propose un moyen pour spécifier au sein d'une même organisation plusieurs politiques de sécurité et définit les concepts suivants :

- **Organisation** : l'entité centrale du modèle OrBAC est l'organisation. Une organisation peut être vue comme un groupe structuré d'entités actives, c'est-à-dire de sujets jouant certains rôles.
- **Sujet** : dans le modèle OrBAC, un sujet peut être soit une entité active, c'est-à-dire un utilisateur, soit une organisation (un service, un département, etc.).
- **Rôle** : le rôle est utilisé pour structurer le lien entre les sujets et les organisations. Comme les sujets jouent des rôles dans des organisations, le modèle OrBAC introduit une relation entre ces entités : la relation "Employer" (ou **Employ**). Soient *org* une organisation, *s* un sujet et *r* un rôle : *Employ(org, s, r)* signifie que *org* emploie le sujet *s* à jouer le rôle *r*.
- **Objet** : un objet représente principalement les entités non actives comme les fichiers, les courriers électroniques, les formulaires imprimés, etc.
- **Vue** : dans la mesure où il est nécessaire de structurer les objets et d'ajouter de nouveaux objets au système, le modèle OrBAC introduit les vues pour regrouper les objets qui satisfont une

propriété commune (par exemple, la vue "dossiers administratifs", "Fiches de salaires", etc.). Dans la mesure où les vues caractérisent la manière dont les objets sont utilisés dans l'organisation, la relation "Utiliser" (ou **Use**) associe ces trois entités. Soient *org* une organisation, *o* un objet et *v* une vue : *Use(org, o, v)* signifie que l'organisation *org* utilise l'objet *o* dans la vue *v*.

- **Action** : les politiques de sécurité spécifient les accès autorisés aux entités passives par des entités actives et régulent les actions opérées dans le système. Dans OrBAC, une action signifie les opérations informatiques comme "lire", "écrire", "supprimer", "envoyer", etc.
- **Activité** : de la même manière que les rôles et les vues sont des abstractions des sujets et des objets, OrBAC définit les activités comme des abstractions des actions. Dans OrBAC, cette entité fait référence à des activités telles que "consulter", "modifier", "nettoyer", "transmettre", etc. Dans la mesure où des organisations différentes peuvent considérer qu'une même action est employée à la réalisation de différentes activités, la relation "Considérer" (ou **Consider**) est utilisée pour associer les entités organisation, action et activité. Soient *org* une organisation, *α* une action et *a* une activité : *Consider(org, α, a)* signifie que l'organisation *org* considère l'action *α* comme faisant partie de l'activité *a*.
- **Contexte** : le contexte est utilisé pour spécifier les circonstances concrètes dans lesquelles les organisations accordent des permissions de réaliser des activités sur des vues. Un contexte peut être vu comme une relation ternaire entre les sujets, les objets et les actions définis dans une organisation. Par conséquent, les entités Organisation, Sujet, Objet, Action et Contexte sont liées par une nouvelle relation appelée "Définir" (ou **Define**). Soient *org* une organisation, *s* un sujet, *α* une action, *o* un objet et *c* un contexte : *Define(org, s, α, o, c)* signifie qu'au sein de l'organisation *org*, le contexte *c* est vrai entre le sujet *s*, l'objet *o* et l'action *α*. Les conditions requises pour qu'un contexte donné soit lié, dans une certaine organisation, aux sujets, aux objets et aux actions sera formellement spécifié par des règles logiques.
- **Permission** : la relation "Permission" correspond à une relation entre les organisations, les rôles, les vues, les activités et les contextes. Soient *org* une organisation, *r* un rôle, *a* une activité, *v* une vue et *c* un contexte : *Permission(org, r, v, a, c)* signifie que l'organisation *org* accorde au rôle *r* la permission de réaliser l'activité *a* sur la vue *v* dans le contexte *c*. Les relations "Interdiction", "Obligation" et "Recommandation" sont définies en suivant le même principe que la "Permission". Au niveau concret, le contrôle d'accès doit permettre de décrire les actions concrètes que réalisent les sujets sur les objets. Dans le but de modéliser des permissions concrètes, le modèle OrBAC introduit la relation "Est_Autorisé" (ou **Is_Permitted**) entre les sujets, les objets et les actions. Soient *s* un sujet, *α* une action et *o* un objet : *Is_Permitted(s, α, o)* signifie que le sujet *s* a la permission de réaliser l'action *α* sur l'objet *o*. Les relations *Est_Interdit* (ou **Is_Prohibited**), *Est_Obligé* (ou **Is_Obligated**) et *Est_Recommandé* (ou **Is_Recommended**) sont définies de la même manière.
  Formellement, la permission peut être exprimée comme suit :

$$\forall org, \forall s, \forall \alpha, \forall o, \forall r, \forall a, \forall v, \forall c$$
$$Permission(org, r, v, a, c) \wedge$$
$$Employ(org, s, r) \wedge$$
$$Consider(org, \alpha, a) \wedge$$
$$Use(org, o, v) \wedge$$
$$Define(org, s, \alpha, o, c) \rightarrow Is\_Permitted(s, \alpha, o)$$

Cette expression signifie que :

Si "dans org, le rôle r peut réaliser l'activité a sur la vue v lorsque le contexte c est Vrai",
et si "dans org, r est affecté au sujet s",
et si "dans org, l'action α fait partie de l'activité a",
et si "dans org, l'objet o fait partie de la vue v",
et si "le contexte c est Vrai pour le triple (org, s, α, o)"
Alors, le sujet s est autorisé à exécuter l'action α sur l'objet o.

OrBAC présente certaines limites pour les systèmes collaboratifs. En effet, comme les règles de sécurité OrBAC ont la forme *Permission (org, r, v, a, c)*, il n'est pas possible de représenter des règles qui impliquent plusieurs organisations indépendantes, voire des sous-organisations autonomes d'un système collaboratif particulier. Il est donc impossible d'associer des autorisations à des utilisateurs appartenant à d'autres organisations partenaires (ou à des sous-organisations). Le cadre PolyOrBAC, basé sur le modèle OrBAC, permet de résoudre cette problématique.

5.3.2. PolyOrBAC

PolyOrBAC est un cadre basé sur le modèle OrBAC et permet d'exprimer des politiques de contrôle d'accès locales pour chaque organisation ainsi que des règles de collaboration impliquant plusieurs organisations. Le but de ce cadre est d'étendre la règle *Permission (org, r, v, a, c)* pour couvrir des accès internes et externes. Afin d'appliquer la collaboration au niveau des services et des ressources, PolyOrBAC est basé sur les mécanismes des services web (SOAP, UDDI et WSDL). Le cadre PolyOrBAC consiste en deux principales phases :

- Négociation des règles d'accès collaboratif
- Accès aux services de collaboration

Nous présentons dans ce qui suit le fonctionnement de ces deux phases.

5.3.2.1. Négociation des règles d'accès collaboratif

L'exemple de la Figure 5.2 illustre le mécanisme de négociation de l'accès collaboratif entre deux organisations : OrgB (fournisseur de données) et OrgA (consommateur de données).

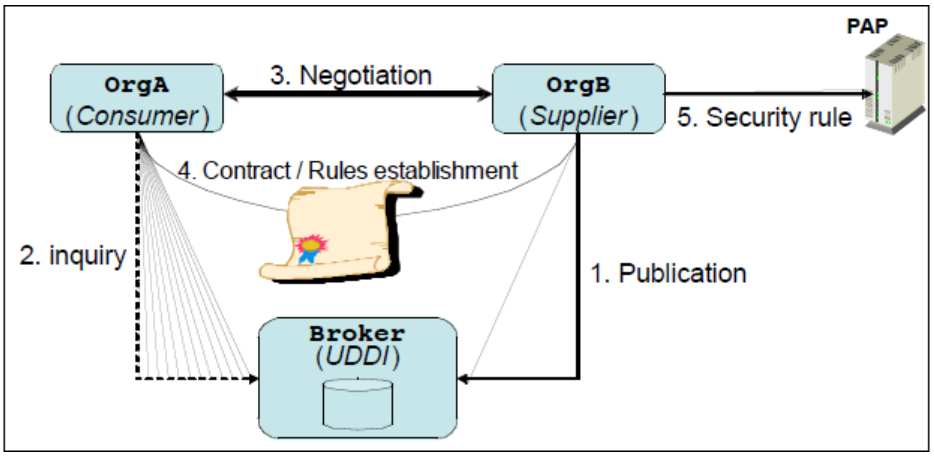

Figure 5.2. PolyOrBAC – négociation des règles d'accès collaboratif [Abou El Kalam, 07]

Cinq principales étapes couvrent cette première phase :

1. **Publication des services par le fournisseur** : une organisation, souhaitant exposer des ressources, commence par développer les services web nécessaires et les déclarer ensuite sur l'annuaire UDDI (Universal Description Discovery and Integration) afin qu'ils soient accessibles à d'autres utilisateurs et organisations externes.
2. **Identification des services par le consommateur** : une fois que les services sont publiés sur l'UDDI, toute organisation intéressée demande l'accès à ces services.
3. **Négociation des règles d'accès et du contrat de collaboration** : les organisations (fournisseur et consommateur) négocient et parviennent à un accord concernant l'utilisation des services web.
4. **Elaboration des règles d'accès et signature du contrat** : les organisations (fournisseur et consommateur) établissent un contrat et définissent conjointement des règles de sécurité concernant l'accès aux services web.
5. **Mise à jour des règles de sécurité par le fournisseur** : les règles de sécurité ainsi définies sont ensuite enregistrées par le fournisseur à travers son PAP (Policy Administration Point).

Admettons que la politique de sécurité de l'organisation OrgB est régie par la règle suivante :

Permission(OrgB, Comptable, Comptes, Consulter, Urgence) ∧
Employ(OrgB, Bob, Comptable) ∧
Consider(OrgB, OuvrirFichiersPDF(), Consulter) ∧
Use(OrgB, WS1, Comptes) ∧
Define(OrgB, Bob, OuvrirFichiersPDF(), WS1, Urgence)
→ Is_Permitted(Bob, OuvrirFichiersPDF(), WS1)

A l'issue de cette première phase, l'organisation OrgB devra ajouter une nouvelle règle à sa politique de sécurité qui représente une dérivation de permissions dans PolyOrBAC :

Permission(OrgB, Comptable, Comptes, Consulter, Urgence) ∧
Employ(OrgB, PartnerA, Comptable) ∧
Consider(OrgB, OuvrirFichiersPDF(), Consulter) ∧
Use(OrgB, WS1, Comptes) ∧
Define(OrgB, PartnerA, OuvrirFichiersPDF(), WS1, Urgence)
→ Is_Permitted(PartnerA, OuvrirFichiersPDF(), WS1)

La règle précédente revient à considérer *PartnerA* comme étant un "sujet virtuel" dans OrgB jouant le rôle Comptable. En réalité, *PartnerA* désigne tout utilisateur de l'organisation OrgA autorisé par OrgA à appliquer l'accord signé avec OrgB pour accéder au web service WS1.

5.3.2.2. Accès aux services de collaboration

PolyOrBAC est basé sur une architecture AAA :

- Authentification (**A**uthentication, en anglais) : cette étape vise à certifier l'identité de l'utilisateur.
- Autorisation (**A**utorization, en anglais) : après s'être authentifié, le serveur basé sur une politique de sécurité décide d'autoriser (ou pas) l'utilisateur d'accéder aux services.
- Traçabilité (**A**ccounting, en anglais) : dans le but d'identifier les responsabilités de chacun, toute action d'un utilisateur authentifié et autorisé est tracée.

Pour illustrer cette deuxième phase, imaginons un utilisateur « Alice » de l'organisation OrgA qui souhaite accéder au service web WS1 de l'organisation OrgB. On distingue entre deux niveaux d'accès : l'accès local qui doit être contrôlé selon la politique de sécurité de l'organisation OrgA et l'accès distant qui doit respecter les accords établis entre OrgA et les autres organisations fournissant les services demandés. Dans ce cadre, Alice devra tout d'abord s'authentifier sur le serveur d'authentification de son organisation OrgA. Ensuite, les mécanismes de sécurité de l'organisation OrgA vérifient si l'accès désiré par Alice est autorisé. Si, dans le cadre de son activité, Alice est autorisée à appeler le service web WS1, l'organisation OrgA lui remet un ticket d'autorisation qu'elle doit présenter au PEP (Policy Enforcement Point) de l'organisation OrgB qui procède à sa vérification conformément à :

- la politique de sécurité OrBAC de l'organisation OrgB, et
- l'accord établi entre les organisations OrgA et OrgB à propos du service web WS1.

Si on reprend l'exemple de notre scénario, les utilisateurs de l'organisation OrgA ayant le droit d'être associés au sujet virtuel *PartnerA* de l'organisation OrgB sont déterminés conformément à la politique de sécurité de l'organisation OrgA. En d'autres termes, c'est l'organisation OrgA qui définit des instances internes telles que (Alice, PartnerA), (Jean, PartnerA), etc. De cette façon, quand Alice est authentifiée et autorisée par son organisation OrgA à être assignée au sujet virtuel PartnerA, un ticket d'autorisation "T1" est généré et accordé à Alice. Le ticket T1 contient les éléments suivants:

- le sujet virtuel joué par Alice : PartnerA
- l'organisation de l'utilisateur Alice : OrgA
- l'identifiant de l'accord entre OrgA et OrgB : ID123
- le service demandé : WS1
- la méthode invoquée : OuvrirFichiersPDF()
- un horodatage pour empêcher les attaques de réponse : 123456789

Quand Alice présente à l'organisation OrgB sa demande (appel de la méthode OuvrirFichiersPDF() du service web WS1) ainsi que son ticket T1 (comme preuve), OrgB extrait les paramètres du ticket T1 et traite la demande. En consultant ses règles de sécurité, OrgB associe le rôle « Comptable » à l'utilisateur virtuel «PartnerA» (représentant Alice dans OrgB) selon la règle *Employ(OrgB, PartnerA, Comptable)*.

5.4. WS-Agreement

Les fournisseurs et les consommateurs de données opèrent dans un contexte dynamique régis par un ensemble de règles, de conditions, d'obligations et de garanties formalisés dans un contrat, le SLA (Service Level Agreement). Il s'agit d'un type de contrat qui spécifie les propriétés de qualité de service QoS (Quality of Service) qui doivent être maintenues par un fournisseur pendant la prestation de service. Ces propriétés définissent les objectifs de niveau de service SLO (Service Level Objectives). Les SLO sont des termes mesurables que l'on surveille pendant toute la durée de vie du contrat. Concernant les techniques de gestion des contrats SLA (création, négociation, re-négociation et surveillance), de nombreuses solutions existent parmi lesquelles, on trouve :

- **WSLA** (Web Service Level Agreement) : c'est une spécification créée par IBM pour la création et la surveillance des contrats SLA. Elle définit un langage permettant aux consommateurs et aux fournisseurs de services de définir et de spécifier des paramètres SLA [Ludwig, 03].

- **WSOL** (Web Service Offering Language) : cette spécification, basée sur le langage XML, permet aux fournisseurs de définir plusieurs niveaux de services pour un même service web dans différentes instances. La particularité de cette approche c'est qu'une offre de service représente une seule classe de services avec une QoS spécifique. Un consommateur peut donc chercher un service souhaité dans le registre des services et sélectionner l'instance qui l'intéresse selon le niveau de service qui convient à ses besoins [Tosic, 02].
- **WS-Agreement** (Web Services Agreement) : c'est une norme du Groupe "The Grid Resource Allocation and Agreement Protocol Working Group". Il s'agit d'une spécification qui définit un protocole et un langage pour négocier, renégocier, créer et surveiller dynamiquement des SLA bilatéraux (entre consommateur et fournisseur) dans les systèmes distribués [Andrieux, 04].

WS-Agreement et son extension WS-Agreement Negotiation [Wäldrich, 11] sont les seules spécifications des SLA standardisées et acceptées par une large communauté [Marino, 18], [Li, 14-b]. En outre, à notre connaissance, ce sont les seules solutions permettant de gérer des SLA bilatéralement négociables. Par conséquent, notre étude est basée sur la spécification WS-Agreement et son extension WS-Agreement Negotiation pour automatiser le protocole de négociation du cadre PolyOrBAC.

5.4.1. Spécification WS-Agreement

WS-Agreement est une spécification qui permet de réaliser des échanges basés sur le langage XML entre les fournisseurs et les consommateurs des services web. Deux types de documents XML existent : le modèle (ou Template, en anglais) pré-rempli par le fournisseur de service et l'accord (ou Agreement en anglais) sur lequel les deux partis se mettent d'accord suite au processus de négociation. Un accord est créé sur la base d'un modèle. Le modèle et l'accord ont la même structure ; il s'agit d'un document XML composé d'un ensemble de sections dont les principales sont représentées dans la Figure 5.3.

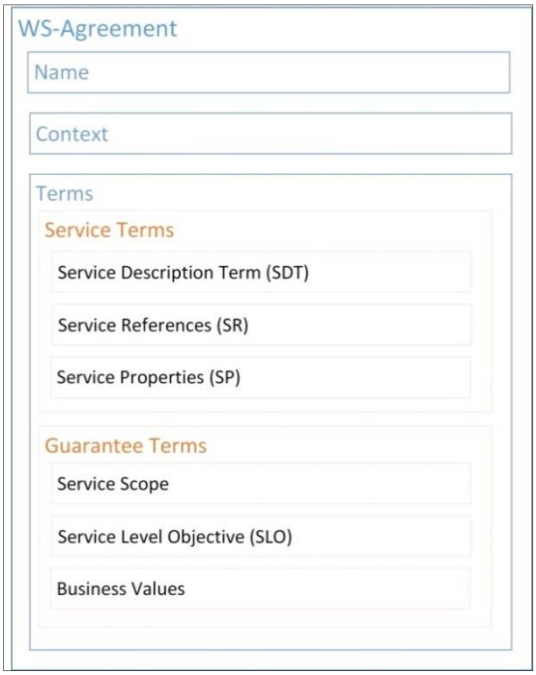

Figure 5.3. Structure des accords et des modèles de la spécification WS-Agreement

Un accord est composé de trois sections :

- **Nom** : cette section contient le nom et l'identifiant de l'accord.
- **Contexte** : le contexte contient des méta-informations sur l'accord telles que les identifiants de l'initiateur de l'accord (généralement c'est le consommateur des services) et du répondeur de l'accord (généralement c'est le fournisseur des services), le nom et l'identifiant du modèle qui a servi de base pour créer l'accord, références vers d'autres accord, la période de validité, etc.
- **Termes** : les termes d'un accord comprennent un ou plusieurs termes de service et zéro ou plusieurs termes de garantie qui définissent les contraintes applicables aux services exposés dans l'accord. La section des termes de service se compose des termes de descriptions de service, des références de service et des propriétés de service. Un terme de garantie de service se compose d'une section de description du scope du service, d'une section de description des objectifs de niveau de service et éventuellement d'une section de description de la valeur commerciale (Business Values) du service.

Voici un exemple d'un accord WS-Agreement :

```
<?xml version="1.0" encoding="UTF-8"?>
<wsag:Agreement xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement"
 AgreementId="polyorbac-agreement-sample">
 <wsag:Name>PolyOrBAC Agreement Sample</wsag:Name>
 <wsag:Context>
  <wsag:AgreementInitiator>OrgA</wsag:AgreementInitiator>
  <wsag:AgreementResponder>OrgB</wsag:AgreementResponder>
  <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
  <wsag:ExpirationTime>2022-12-30T12:00:00</wsag:ExpirationTime>
  <wsag:TemplateId>ws1-template-01</wsag:TemplateId>
  <wsag:TemplateName>ws1-template</wsag:TemplateId>
 </wsag:Context>
 <wsag:Terms>
  <wsag:All>
   <wsag:ServiceDescriptionTerms Name="SDT" ServiceName="WS1"/>
   <wsag:ServiceProperties Name="ServiceProperties" ServiceName="WS1">
    <wsag:VariableSet>
     <wsag:Variable Name="availability" Metric="xs:double">
      <wsag:Location>metric1</wsag:Location>
     </wsag:Variable>
    </wsag:VariableSet>
   </wsag:ServiceProperties>
   <wsag:GuaranteeTerm Name="GT-availability">
    <wsag:ServiceScope ServiceName="ServiceName"/>
    <wsag:ServiceLevelObjective>
     <wsag:KPITarget>
      <wsag:KPIName>AVAILABILITY</wsag:KPIName>
      <wsag:CustomServiceLevel>
       {"constraint" : "availability BETWEEN (0.99, 1)"}
      </wsag:CustomServiceLevel>
     </wsag:KPITarget>
    </wsag:ServiceLevelObjective>
   </wsag:GuaranteeTerm>
  </wsag:All>
 </wsag:Terms>
</wsag:Agreement>
```

Nous détaillons dans les sous-sections suivantes les principaux éléments d'un accord WS-Agreement en précisant la définition selon la spécification officielle [Andrieux, 04].

5.4.1.1. Structure d'un accord WS-Agreement

Voici la spécification d'un modèle WS-Agreement :

```
<wsag:Template TemplateId="xs:string">
       <wsag:Name>xs:string</wsag:Name> ?
       <wsag:Context>
              wsag:AgreementContextType
       </wsag:Context>
       <wsag:Terms>
              wsag:TermCompositorType
       </wsag:Terms>
</wsag:Template>
```

Un modèle, encapsulé dans la balise `<wsag:Template/>`, est identifié par un identifiant unique (`TemplateId`) et un nom (`wsag:Name`) et contient un contexte (`<wsag:Context/>`) et une collection de termes (`<wsag:Terms/>`).

Un accord, qui est une instance d'un modèle, suit la même spécification :

```
<wsag:Agreement AgreementId="xs:string">
       <wsag:Name>xs:string</wsag:Name> ?
       <wsag:Context>
              wsag:AgreementContextType
       </wsag:Context>
       <wsag:Terms>
              wsag:TermCompositorType
       </wsag:Terms>
</wsag:Agreement>
```

Un accord, encapsulé dans la balise `<wsag:Agreement/>`, est identifié par un identifiant unique (`AgreementId`) et un nom (`wsag:Name`) et contient un contexte (`<wsag:Context/>`) et une collection de termes (`<wsag:Terms/>`).

5.4.1.1.1. Contexte d'un accord WS-Agreement

Le contexte décrit certaines métadonnées sur l'accord. Voici sa spécification :

```
<wsag:Context xs:anyAttribute>
       <wsag:AgreementInitiator>xs:anyType</wsag:AgreementInitiator> ?
       <wsag:AgreementResponder>xs:anyType</wsag:AgreementResponder> ?
       <wsag:ServiceProvider>wsag:AgreementRoleType</wsag:ServiceProvider>
       <wsag:ExpirationTime>xs:DateTime</wsag:ExpirationTime> ?
       <wsag:TemplateId>xs:string</wsag:TemplateId> ?
       <wsag:TemplateName>xs:string</wsag:TemplateName> ?
       <xs:any/> *
</wsag:Context>
```

Le contexte contient les informations suivantes :

- `wsag:AgreementInitiator` : l'initiateur de la demande de création de l'accord. Selon la spécification WS-Agreement, l'initiateur peut être le consommateur ou le fournisseur du service.

- `wsag:AgreementResponder` : le répondant à l'accord, il s'agit de l'entité qui répond à la demande de création de l'accord. Si le consommateur du service est l'initiateur, le fournisseur du service est le répondant et inversement.
- `wsag:ServiceProvider` : cet élément identifie le fournisseur du service. Il prend une des deux valeurs : `AgreementInitiator` ou `AgreementResponder` (par défaut, il prend la valeur `AgreementResponder`).
- `wsag:ExpirationTime` : la date d'expiration de l'accord.
- `wsag:TemplateId` : l'identifiant du modèle utilisé pour créer l'accord.
- `wsag:TemplateName` : le nom du modèle utilisé pour créer l'accord. Si on prend l'exemple du cadre WSAG4J, qui est une implémentation de la spécification WS-Agreement, un modèle est identifié par un identifiant et un nom uniques dans l'Agreement Factory. Si on prend l'exemple de SLA-Framework, qui est une autre implémentation de la spécification WS-Agreement, un modèle est identifié par un identifiant unique.

Voici un exemple du contexte d'un accord:

```
<wsag:Context>
  <wsag:AgreementInitiator>OrgA</wsag:AgreementInitiator>
  <wsag:AgreementResponder>OrgB</wsag:AgreementResponder>
  <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
  <wsag:ExpirationTime>2022-12-30T12:00:00</wsag:ExpirationTime>
  <wsag:TemplateId>ws1-template-01</wsag:TemplateId>
  <wsag:TemplateName>ws1-template</wsag:TemplateId>
</wsag:Context>
```

5.4.1.1.2. Termes d'un accord WS-Agreement

Les termes d'un accord comprennent un ou plusieurs termes de définition de service et zéro ou plusieurs termes de garantie regroupés à l'aide d'opérateurs de regroupement logique. Trois types de termes permettent de décrire un service : termes de description du service, des références du service et des propriétés du service. La section optionnelle des termes de garantie détient les contraintes applicables aux services exposés dans l'accord (Services Scopes, SLO et Business Value).

5.4.1.1.2.1. Termes de description de service (SDT)

Les termes de description de service décrivent les services exposés par le fournisseur. Voici sa définition :

```
<wsag:ServiceDescriptionTerm
      wsag:Name="xs:string" wsag:ServiceName="xs:string">

      <xs:any> ... </xs:any>
</wsag:ServiceDescriptionTerm> +
```

Une implémentation peut fournir des fonctionnalités supplémentaires pour gérer les termes de description de service. Par exemple, le SDT peut être rempli par le système avec les informations nécessaires sur les ressources allouées (URL, adresse IP, etc.).

Voici un exemple de la section `ServiceDescriptionTerm` :

```
<wsag:ServiceDescriptionTerms Name="SDT-WS1" ServiceName="WS1"/>
<wsag:ServiceDescriptionTerms Name="SDT-WS2" ServiceName="WS2"/>
```

5.4.1.1.2.2. Références de Service (SR)

Une référence de service pointe vers un service. Ainsi, si le service fourni dans l'accord est un service externe, il peut être référencé dans cette section. De cette façon, l'URL, les identifiants et tout ce qui est associé à un attribut `ServiceName` peut être déclaré dans cette section. Voici sa définition :

```
<wsag:ServiceReference
      wsag:Name="xs:string" wsag:ServiceName="xs:string">

      <xs:any> ... </xs:any>
</wsag:ServiceReference> +
```

Voici un exemple de la section `ServiceReference` :

```
<wsag:ServiceReference wsag:Name="CountingRef" wsag:ServiceName="CountingService">
  <wsa:EndPointReference>
      <wsa:Address>/Counting</wsa:Address>
  </wsa:EndPointReference>
</wsag:ServiceReference>
<wsag:ServiceReference wsag:Name="Monitoring" wsag:ServiceName="MonitoringService">
      <wsa:EndPointReference>
            <wsa:Address>/Monitoring</wsa:Address>
      </wsa:EndPointReference>
</wsag:ServiceReference>
```

5.4.1.1.2.3. Propriétés de Service (SP)

Les propriétés de service sont utilisées pour définir des propriétés mesurables et exposées associées à un service telles que le temps de réponse, le débit, etc. Voici sa définition :

```
<wsag:ServiceProperties wsag:Name="xs:string" wsag:ServiceName="xs:string">
      <wsag:VariableSet>
            <wsag:Variable wsag:Name="xs:string" wsag:Metric="xs:URI">
                  <wsag:Location>xs:string</wsag:Location>
            </wsag:Variable> *
      </wsag:VariableSet>
</wsag:ServiceProperties> +
```

Les propriétés de service correspondent à un ensemble de variables utilisées dans les contraintes des Termes de Garantie. Si, par exemple, une contrainte est : *uptime* > *90*, elle pourrait avoir deux propriétés de service: *ActualUptime* et *DesiredUptime* ; la contrainte sera alors *ActualUptime* > *DesiredUptime*. Voici un exemple de la section `ServiceProperties` :

```
<wsag:ServiceProperties Name="ServiceProperties" ServiceName="WS1">
<wsag:VariableSet>
 <wsag:Variable Name="availability" Metric="xs:double">
      <wsag:Location>metric1</wsag:Location>
 </wsag:Variable>
 <wsag:Variable wsag:Name="service_state" wsag:Metric="xs:string">
      <wsag:Location>metric1</wsag:Location>
 </wsag:Variable>
</wsag:VariableSet>
</wsag:ServiceProperties>
```

5.4.1.1.2.4. Termes de Garantie (GT)

La section optionnelle des termes de garantie détient les contraintes applicables aux services exposés dans l'accord. Voici sa définition:

```
<wsag:GuaranteeTerm Name="xs:string" Obligated="wsag:ServiceRoleType">
      <wsag:ServiceScope ServiceName="xs:string">
            xs:any ?
      </wsag:ServiceScope> *
      <wsag:QualifyingCondition> xs:anyType </wsag:QualifyingCondition> ?
      <wsag:ServiceLevelObjective>
            <wsag:KPITarget>
                  <wsag:KPIName>xs:string</wsag:KPIName>
                  <wsag:CustomServiceLevel>xs:any</wsag:CustomServiceLevel>
            </wsag:KPITarget>
       |
            <wsag:CustomServiceLevel> xs:anyType </wsag:CustomServiceLevel>
      </wsag:ServiceLevelObjective>
      <wsag:BusinessValueList>
            <wsag:Importance> xs:integer </wsag:Importance> ?
            <wsag:Penalty>
                  <wsag:AssessmentInterval>
                        <wsag:TimeInterval>xs:duration</wsag:TimeInterval> |
                        <wsag:Count>xs:positiveInteger</wsag:Count>
                  </wsag:AssessmentInterval>
                  <wsag:ValueUnit>xs:string</wsag:ValueUnit>?
                  <wsag:ValueExpression>xs:anyType</wsag:ValueExpr>
            </wsag:Penalty> *
            <wsag:Reward>
                  <wsag:AssessmentInterval>
                        <wsag:TimeInterval>xs:duration</wsag:TimeInterval> |
                        <wsag:Count>xs:positiveInteger</wsag:Count>
                  </wsag:AssessmentInterval>
                  <wsag:ValueUnit>xs:string</wsag:ValueUnit>?
                  <wsag:ValueExpression>xs:anyType</wsag:ValueExpr>
            </wsag:Reward> *
            <wsag:Preference>
                  <wsag:ServiceTermReference>xs:string
</wsag:ServiceTermReference> *
                  <wsag:Utility>xs:float</wsag:Utility> *
            </wsag:Preference> ?
            <wsag:CustomBusinessValue>xs:anyType</wsag:CustomBusinessValue> *
      </wsag:BusinessValueList>
</wsag:GuaranteeTerm>
```

Cette section comprend de nombreux éléments :

- L'attribut `Name` de l'élément `GuaranteeTerm` : l'attribut `Name` est obligatoire ; il représente le nom donné à une garantie. Puisqu'un accord peut englober plusieurs termes de garantie, chaque terme devra avoir un nom unique.
- L'attribut `Obligated` de l'élément `GuaranteeTerm` : cet attribut définit quelle partie (consommateur ou fournisseur) prend l'obligation. Le `wsag:ServiceRoleType` peut être `ServiceConsumer` ou `ServiceProvider`. Par défaut c'est `ServiceProvider`.
- L'élément `ServiceScope` : un terme de garantie peut avoir zéro ou plusieurs `ServiceScope`. Un scope (ou portée) de service décrit à quel élément des services exposés le terme de garantie s'applique. Il contient un attribut `ServiceName` et toute autre structure XML (`xs:any ?`) décrivant une sous-structure d'un service auquel s'applique la portée. Par exemple, une garantie de

performances peut ne s'appliquer qu'à une seule opération d'un service web à un point final (end-point) particulier.

- L'attribut `ServiceName` de l'élément `ServiceScope` : le nom du service auquel le terme de garantie fait référence. Un scope de service s'applique à un, et un seul, service.
- L'élément `ServiceLevelObjective` : cet élément spécifie un objectif de niveau de service dans un terme de garantie et contient un élément de type `KPITarget` ou `CustomServiceLevel` :
  - `KPITarget`: cet élément définit l'objectif de niveau de service comme l'expression d'une cible d'un indicateur de performance clé associé au service.
  - `CustomServiceLevel` : cet élément, de type `xs:anyType`, peut être personnalisé en utilisant une expression spécifique au domaine ou un langage d'assertion qui peut être conçu indépendamment de la spécification WS-Agreement.
- L'élément `BusinessValueList` : à chaque SLO est associée une liste de valeurs commerciales, chacune exprime un aspect de valeur de l'objectif. Le concept derrière cela est qu'une violation d'un terme de garantie peut entraîner une sanction commerciale. En revanche, un terme de garantie accompli peut impliquer une récompense commerciale. Dans l'exemple ci-après, une violation du SLO toutes les 100 invocations au service entraîne une pénalisation de 10€.

Voici un exemple de la section `GuaranteeTerm` :

```
<wsag:GuaranteeTerm Name="GT-ResponseTime">
  <wsag:ServiceScope ServiceName="web-service-ping"/>
  <wsag:ServiceLevelObjective>
    <wsag:KPITarget>
      <wsag:KPIName>Uptime</wsag:KPIName>
      <wsag:CustomServiceLevel>
          {"constraint" : "Uptime BETWEEN (90, 100)"}
      </wsag:CustomServiceLevel>
    </wsag:KPITarget>
  </wsag:ServiceLevelObjective>
  <wsag:BusinessValueList>
    <wsag:Importante>3</wsag:Importante>
    <wsag:Penalty>
      <wsag:AssessmentInterval>
        <wsag:Count>100</wsag:Count>
      </wsag:AssessmentInterval>
      <wsag:ValueUnit>EUR</wsag:ValueUnit>
      <wsag:ValueExpression>10</wsag:ValueExpression>
    </wsag:Penalty>
  </wsag:BusinessValueList>
</wsag:GuaranteeTerm>
```

5.4.1.2. Protocole de négociation WS-Agreement

Par définition, la "négociation" est le processus de discussion entre deux ou plusieurs parties dans le but d'arriver à un accord commun défendant les intérêts de chacun. Par analogie, la négociation d'accès doit permettre à des fournisseurs et des consommateurs de services web de négocier un ensemble de termes dans le but de conclure un accord commun cadrant toutes les modalités d'accès. L'automatisation des négociations exige la formalisation de la définition de chaque terme du contrat d'accès pour qu'il soit compréhensible par la machine. De nombreux travaux ont été réalisés ces dernières années pour automatiser les négociations des contrats dans différents domaines tels que les objets connectés ([Marino,

18], [Li, 19-a], [Li, 19-b]), les plateformes du Cloud Computing ([Shojaiemehra, 19], [Scoca, 17], [Labidi, 17], [Baig, 17]) ainsi que les environnements distribués ([Li, 14-b], [Tseng, 16], [Castro, 15], [Coutinho, 14]).

Le modèle de négociation WS-Agreement, représenté dans la Figure 5.4, se compose de trois couches avec une séparation claire entre ces couches :

- **la couche de négociation** : cette couche fournit un protocole et un langage pour négocier des offres d'accord. Le processus de négociation comprend l'échange d'offres et de contre-offres de négociation. Une offre de négociation indique la volonté des deux parties de conclure un accord ultérieur. A ce stade, aucun engagement n'est conclu.
- **la couche de l'accord** : cette couche fournit un protocole et un langage permettant d'assurer les fonctionnalités de base pour créer et surveiller les accords.
- **la couche de service** : les services proposés par le fournisseur de données sont définis au niveau de cette couche. Un service défini par un accord peut comprendre plusieurs autres services. L'exécution d'un service est régie par la couche accord.

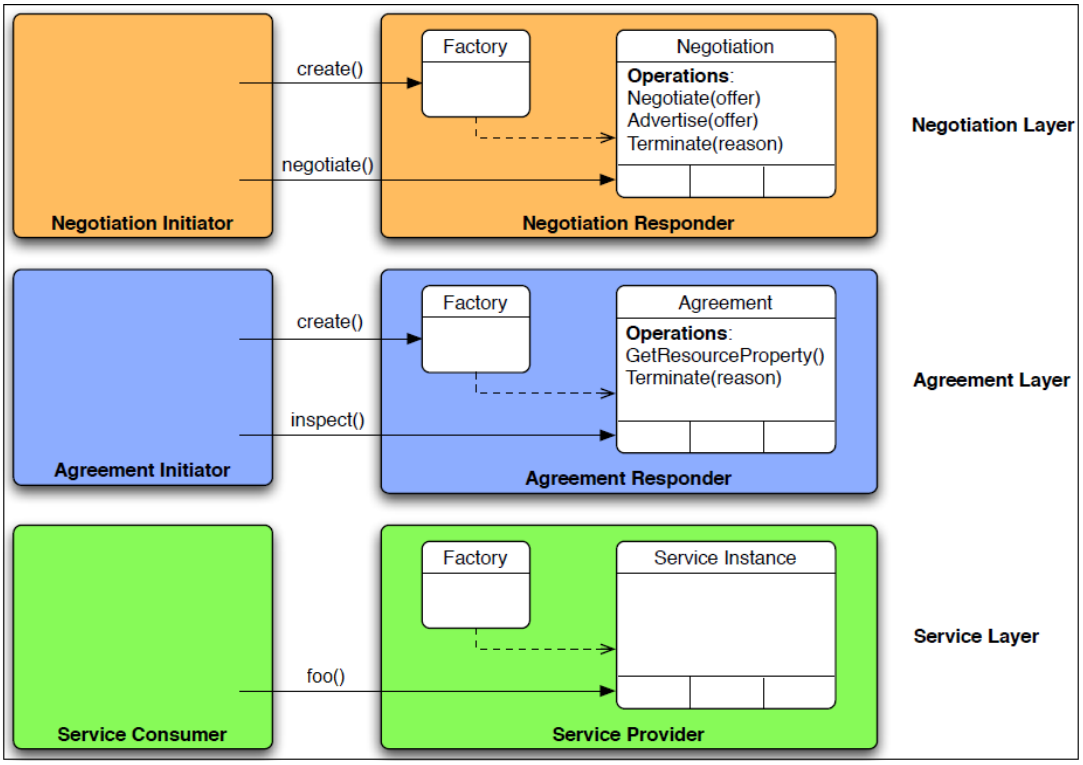

Figure 5.4. Modèle de négociation WS-Agreement [Wäldrich, 11]

Le protocole de négociation WS-Agreement WSANP (WS-Agreement Negotiation Protocol) est aujourd'hui le standard utilisé en matière de négociation SLA pour les services web pour de nombreux projets [Marino, 18], [Li, 19-b]. Ce protocole définit les services et les opérations requis pour créer, négocier, renégocier et surveiller des SLA. La Figure 5.5 illustre l'implémentation du protocole WSANP dans le cadre WSAG4J que nous présenterons dans la section suivante.

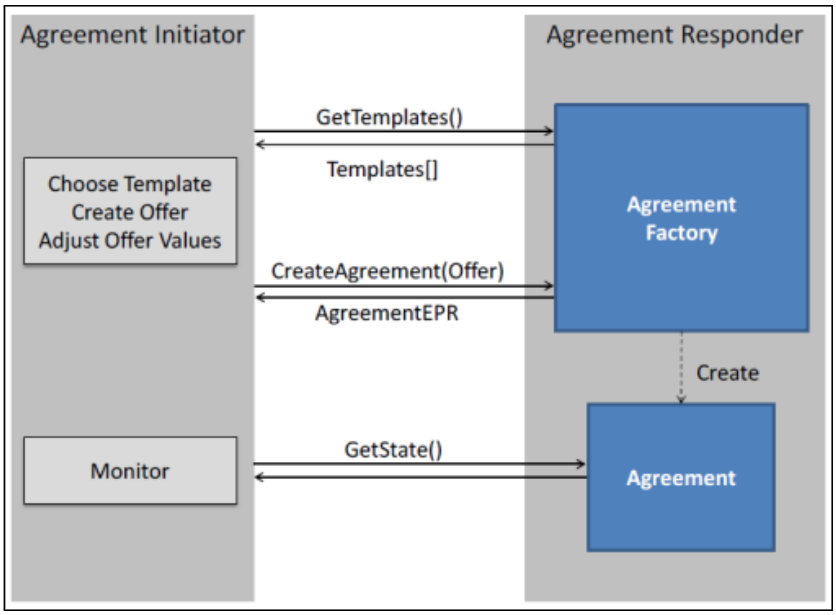

Figure 5.5. Protocole de négociation WS-Agreement [wsag4j, 12]

L'initiateur de l'accord (le consommateur des services), récupère auprès du répondeur de l'accord (le fournisseur des services) les différents modèles de contrats disponibles afin d'en sélectionner celui qui lui convient. L'initiateur de l'accord établit une offre en complétant le modèle choisi et demande la création de l'accord au répondeur de l'accord. Ce dernier dispose de deux types de port[46] : le type de port "Agreement Factory", chargé de créer des accords et d'instancier le service associé avec la qualité de service convenue, et le type de port "Agreement State", chargé du contrôle de la conformité des accords et des services associés. Une fois que l'accord est créé, et pendant toute sa durée de vie, le consommateur de services peut récupérer son état grâce aux différentes métriques mises à sa disposition par le fournisseur de services.

La Figure 5.6 montre un exemple de négociation bilatérale. Dans cet exemple, l'initiateur de la négociation démarre le processus de négociation en récupérant les modèles (appel de la méthode *getTemplates*) fournis par le répondeur. Un modèle comprend la description du service et de ses garanties, ainsi qu'un ensemble d'options mises à disposition du consommateur (par exemple, pour le temps de réponse moyen d'un service, le fournisseur peut proposer deux niveaux : High, Medium). Le prix de l'offre de service dépend alors du type et du nombre des ressources informatiques sélectionnées et des niveaux de service choisis par le consommateur. La négociation est ensuite initiée par l'initiateur en envoyant une demande d'initiation de négociation à la *NegotiationFactory* du répondeur (appel de la méthode *initiateNegotiation*). Cette demande inclut une référence du point de terminaison *InitiatorEPR* (End Point Reference). L'initiateur informe ensuite le répondeur des offres qu'il est prêt à négocier en appelant la méthode *advertise* à laquelle répond le répondeur en envoyant des offres à l'initiateur. Après plusieurs cycles de négociation (boucle d'offres et de contre-offres), l'initiateur peut décider de créer un accord basé sur l'une des offres négociées en appelant la méthode *createAgreement* du répondeur qui s'occupe alors de la création de l'accord relatif au point de terminaison de référence (*AgreementEPR*).

[46] Un type de port est un ensemble abstrait d'opérations prises en charge par un ou plusieurs points de terminaison (end-points).

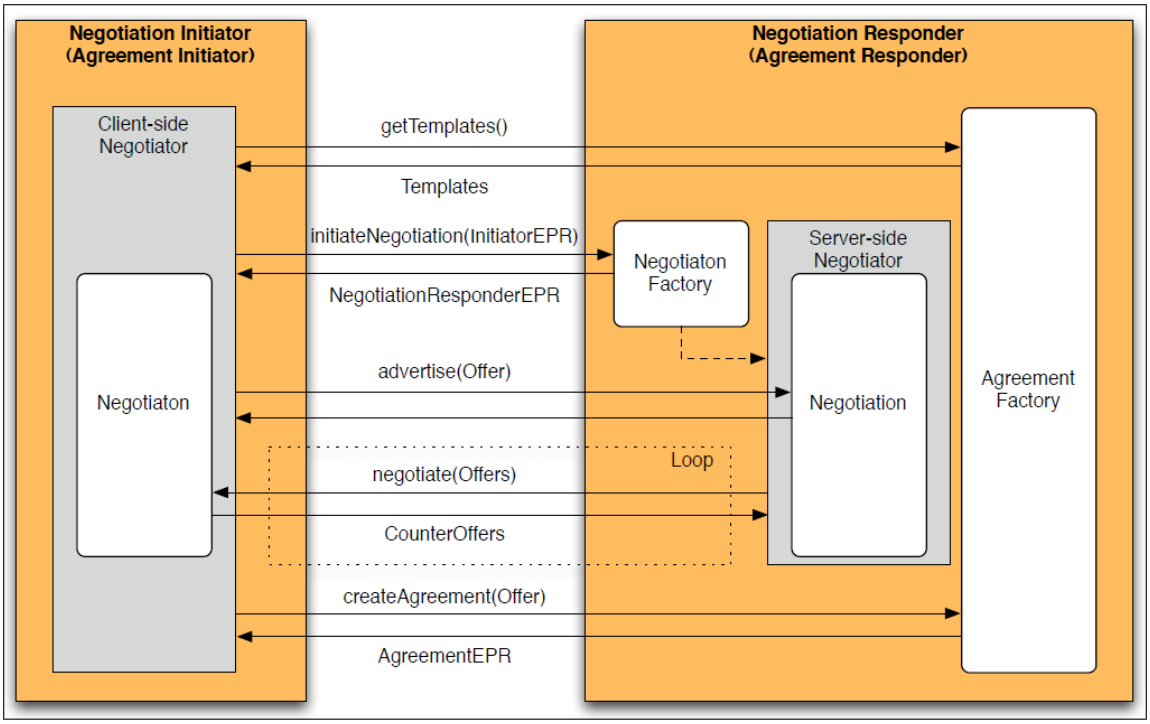

Figure 5.6. Exemple de négociation bilatérale WS-Agreement [Battre, 10]

5.4.2. Implémentations WS-Agreement

5.4.2.1. WSAG4J (WS-Agreement for Java)

WSAG4J [wsag4j, 12] est une implémentation générique du protocole WS-Agreement. Elle prend en charge des fonctionnalités communes pour créer et surveiller les accords de manière générique et permet aux utilisateurs de créer et de déployer rapidement des services basés sur WS-Agreement. WSAG4J implémente le type de port "Agreement Factory" pour la création des accords et le type de port "Agreement State" pour surveiller les états des accords existants. WSAG4J suit une approche déclarative pour soutenir et gérer l'ensemble du cycle de vie d'un accord : de la définition d'un modèle d'accord, en passant par le déploiement des modèles dans les "Factories" et jusqu'à la gestion de l'accord. Voici certaines fonctionnalités assurées par le cadre WSAG4J :

- **Conception des modèles d'accords** : WSAG4J aide les développeurs à concevoir des modèles SLA en permettant de publier des modèles sous forme de fichiers XML. Des exemples fournis par WSAG4J peuvent être modifiés afin de simplifier la conception des modèles.
- **Validation des offres d'accords** : WS-Agreement définit un ensemble de contraintes qui doivent être validées avant la création d'un accord. Ces contraintes définissent comment une offre d'un accord valide basée sur un modèle spécifique est construite. Ces contraintes comprennent, d'une part, la structure de l'offre de l'accord (par exemple, les termes de description du service), et d'autre part, la définition des valeurs concrètes qu'un élément spécifique de l'offre peut prendre (par exemple le temps minimal et maximal pour répondre à une requête).
- **Évaluation des garanties d'accord** : WSAG4J fournit également des moyens automatiques pour l'évaluation des garanties. À l'instar de la validation des contraintes de création, la définition des

conditions de garantie sert de base à l'évaluation de la garantie. WSAG4J propose certaines bonnes pratiques qui doivent être respectées afin d'évaluer avec succès les conditions de garantie.

5.4.2.2. SLA-Framework

SLA-Framework [SLA-Framework, 16] est une autre implémentation de la spécification WS-Agreement. Il s'agit d'un projet libre qui permet de gérer le cycle de vie des accords SLA (la création, le suivi et la résiliation des accords). Actuellement en version 1.1, ce cadre ne prend pas en charge les boucles de négociations (une suite d'offres et de contre-offres). Pour surmonter cette limite, il faudra améliorer cette fonctionnalité du cadre ou, pour l'utiliser tel quel, gérer l'historique des échanges entre les fournisseurs et les consommateurs des services web. Pour ne pas rester bloqué à ce stade, on peut considérer qu'une offre faite par le consommateur doit être acceptée ou refusée par le fournisseur et se dispenser ainsi de la possibilité des contre-offres. Le noyau SLA-Framework, SLA Manager, fournit les fonctionnalités suivantes :

- Un langage et un protocole pour définir et annoncer les capacités des fournisseurs de services dans les formulaires des modèles SLA.
- La création des accords basés sur les modèles.
- Le suivi du respect des accords lors de l'exécution.

5.5. Extension du cadre PolyOrBAC

Pour automatiser le mécanisme de négociation des accords d'accès dans le cadre PolyOrBAC, nous proposons dans la Figure 5.7 un nouveau protocole compatible avec le modèle WSANP.

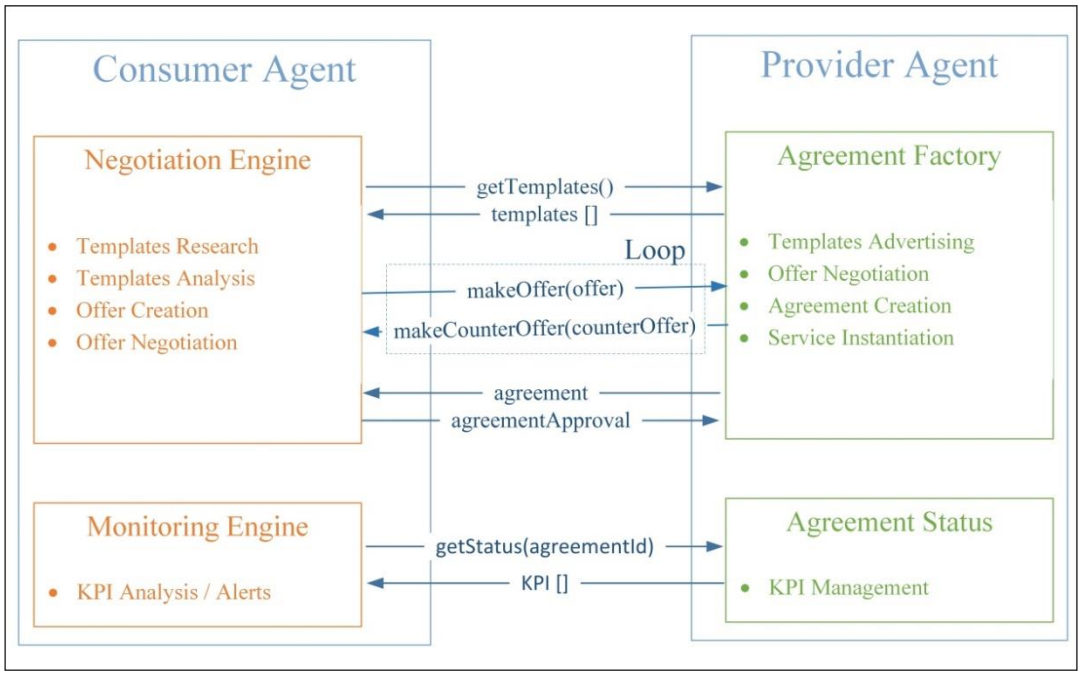

Figure 5.7. Protocole de négociation des accords d'accès dans PolyOrBAC [Talha, 21-a]

Un fournisseur de services web modélise et publie des modèles conformément à la spécification WS-Agreement dans lesquels il décrit les services qu'il souhaite exposer. Les négociations des règles d'accès et des autres conditions de l'accord sont effectuées entre les agents représentant le fournisseur (Provider Agent) et le consommateur (Consumer Agent) en suivant les étapes suivantes :

1. L'agent consommateur identifie les services dont la description correspond à ce qui l'intéresse et récupère auprès de l'agent fournisseur la liste des modèles en appelant sa méthode *getTemplates()*.
2. L'agent consommateur analyse la liste des modèles et en choisit un à partir duquel l'accord final sera créé. Le modèle est déjà initialisé par l'agent fournisseur, l'agent consommateur complète les champs qui le concernent, adapte les règles d'accès aux services qu'il souhaite et propose les valeurs souhaitées pour chaque SLO. L'offre ainsi construite est envoyée à l'agent fournisseur en faisant appel à la méthode *makeOffer(offer)*.
3. A la réception de l'offre, un processus de validation est initié par l'agent fournisseur. Ce processus consiste à vérifier les valeurs saisies par l'agent consommateur en appliquant les contraintes de création de l'accord prédéfinies dans le modèle et en validant les options SLO souhaitées par l'agent consommateur :
    - Si l'offre est validée, l'agent fournisseur crée l'accord, le signe et l'envoie à l'agent consommateur pour approbation.
    - Sinon, il adapte les valeurs souhaitées par l'agent consommateur et lui fait une contre-offre en appelant la méthode *makeCounterOffer(counterOffer)*.
4. A la réception de la contre-offre, l'agent consommateur peut la refuser (fin du processus de négociation) ou l'adapter en créant une nouvelle offre. Dans ce dernier cas, la nouvelle offre est envoyée à l'agent fournisseur en faisant appel à la méthode *makeOffer(offer)*. Les étapes 3 et 4 représentent une boucle d'offres et de contre-offres entre les deux agents. Le nombre d'itérations dans la boucle de négociation étant limité, si aucun accord n'est conclu à la fin de la boucle, le processus de négociation échoue.
5. A la réception de l'accord signé, l'agent consommateur signe sa part et transmet son approbation à l'agent fournisseur.
6. Une fois que l'accord est créé, signé et approuvé, l'agent fournisseur met à jour la politique de sécurité du fournisseur de services en fonction des règles d'accès négociées dans l'accord. Il fournit également à l'agent consommateur les différents KPI (Key Performance Indicators) nécessaires à la surveillance de l'accord.

Pour illustrer ce protocole de négociation, considérons le scénario suivant : «*Une organisation OrgB dispose d'une Fiche de Salaires contenant l'historique des salaires de tous ses employés. Cette fiche peut être consultée par un directeur de l'organisation lors d'un contrôle financier ou mise à jour par un comptable lors du transfert des salaires. Concrètement, pour consulter les comptes de l'entreprise, John, un directeur de l'entreprise, peut appeler la méthode "get" du service web SalaryWS. Bob, un comptable de l'entreprise, peut appeler la méthode "put" de ce même service web SalaryWS pour mettre à jour la fiche. Pour des fins réglementaires, après chaque modification de la Fiche de Salaires, un email doit être adressé, dans un délai maximum de 24 heures, aux salariés concernés par le changement.* ». Ce scénario est illustré dans la Figure 5.8.

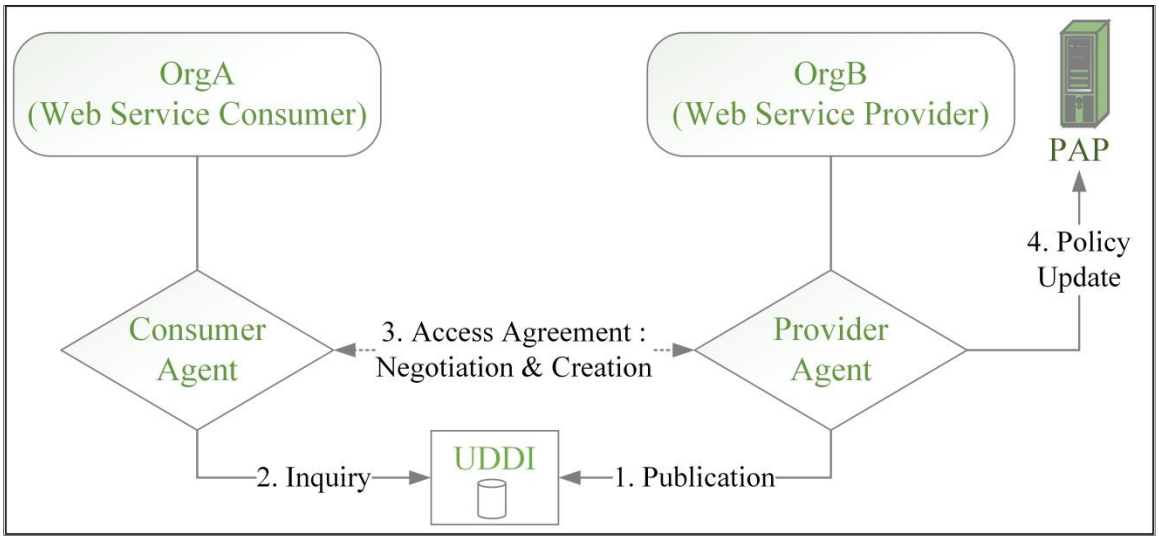

Figure 5.8. PolyOrBAC - Négociation des règles d'accès collaboratif [Talha, 21-a]

Dans OrBAC, ce scénario peut être implémenté en utilisant les règles suivantes :

Permission (OrgB, Directeur, Fiche de Salaires, Consulter, Contrôle Financier) ∧
Employ (OrgB, John, Directeur) ∧
Consider (OrgB, get, Consulter) ∧
Use (OrgB, SalaryWS, Fiche de Salaires) ∧
Define (OrgB, John, get, SalaryWS, Contrôle Financier) ∧
→ Is_Permitted (John, get, SalaryWS)

Permission (OrgB, Comptable, Fiche de Salaires, Mettre à Jour, Transfert de Salaires) ∧
Employ (OrgB, Bob, Comptable) ∧
Consider (OrgB, put, Mettre à Jour) ∧
Use (OrgB, SalaryWS, Fiche de Salaires) ∧
Define (OrgB, Bob, put, SalaryWS, Transfert de Salaires) ∧
→ Is_Permitted (Bob, put, SalaryWS) ∧
→ Is_Obligated (Bob, sendEmail, SalaryWS)

Avec :

OrgB ∈ Organizations
{Comptable, Directeur} ∈ Roles
{Consulter, Mettre à Jour} ∈ Activities
{Fiche de Salaires} ∈ Views
{Contrôle Financier, Transfert de Salaires} ∈ Contexts
{John, Bob} ∈ Subjects
{get, put, sendEmail}∈ Action
{SalaryWS} ∈ Objects

Pour illustrer l'activité collaborative, complétons notre scénario: «*Dans le cadre d'un contrat de sous-traitance, Alice de l'organisation OrgA peut consulter la fiche de salaires de l'organisation OrgB. Pour ce faire, elle doit appeler la méthode "get" du service web SalaryWS. Suite à l'externalisation du transfert de salaire, Alice peut également mettre à jour la fiche de salaires de l'organisation OrgB via la méthode "put" du service web SalaryWS* ».

Selon PolyOrBAC, l'appel de la méthode "get" peut être réalisé via :

- un accord d'accès valide négocié entre OrgA et OrgB.
- un sujet virtuel, par exemple PartnerA1, ayant le rôle de Directeur.
- une nouvelle règle collaborative ajoutée à la politique de sécurité de l'organisation OrgB :

Permission (OrgB, Directeur, Fiche de Salaires, Consulter, Contrôle Financier) ∧
Employ (OrgB, PartnertA1, Directeur) ∧
Consider (OrgB, get, Consulter) ∧
Use (OrgB, SalaryWS, Fiche de Salaires) ∧
Define (OrgB, PartnertA1, get, SalaryWS, Contrôle Financier) ∧
→ Is_Permitted (PartnertA1, get, SalaryWS)

- un Ticket T1, chiffré par OrgA, qu'Alice doit présenter avec les informations suivantes :
  - ◊ l'organisation de l'utilisateur Alice : ***OrgA***
  - ◊ le sujet virtuel associé à Alice : ***PartnerA1***
  - ◊ l'identifiant de l'accord entre OrgA et OrgB : ***ID123456***
  - ◊ le service demandé : ***SalaryWS***
  - ◊ l'action demandée : ***get***
  - ◊ un horodatage pour empêcher les attaques de réponse : ***1603558484***

L'appel de la méthode "put" peut se faire grâce à :

- un accord d'accès valide négocié entre OrgA et OrgB.
- un sujet virtuel, par exemple PartnerA2, ayant le rôle de comptable.
- une nouvelle règle collaborative ajoutée à la politique de sécurité de l'organisation OrgB :

Permission (OrgB, Comptable, Fiche de Salaires, Mettre à Jour, Transfert de Salaires) ∧
Employ (OrgB, PartnertA2, Comptable) ∧
Consider (OrgB, put, Mettre à Jour) ∧
Use (OrgB, SalaryWS, Fiche de Salaires) ∧
Define (OrgB, PartnertA2, put, SalaryWS, Transfert de Salaires) ∧
→ Is_Permitted (PartnertA2, put, SalaryWS) ∧
→ Is_Obligated (PartnertA2, sendEmail, SalaryWS)

- un Ticket T2, chiffré par OrgA, qu'Alice doit présenter avec les informations suivantes :
  - ◊ l'organisation de l'utilisateur Alice : ***OrgA***
  - ◊ le sujet virtuel associé à Alice : ***PartnerA2***
  - ◊ l'identifiant de l'accord entre OrgA et OrgB : ***ID123456***
  - ◊ le service demandé : ***SalaryWS***
  - ◊ l'action demandée : ***put***
  - ◊ un horodatage pour empêcher les attaques de *réponse :* ***1603902862***

L'implémentation du protocole illustré dans la Figure 5.7 nécessite la négociation des accords d'accès entre les agents (Consumer Agent et Provider Agent). Pour cela, nous avons choisi SLA-Framework qui est une implémentation de la spécification WS-Agreement. Toutefois, SLA-Framework ne supporte pas l'expression des règles de sécurité et nécessite, par conséquent, une extension pour pouvoir couvrir les besoins du cadre PolyOrBAC. Dans la section suivante, nous expliquons comment nous avons étendu SLA-Framework et reviendrons ensuite sur l'implémentation de notre étude de cas.

5.6. Extension de SLA-Framework

Le cadre PolyOrBAC spécifie des règles de sécurité via des automates temporisés dans lesquelles :

- Les autorisations, qui représentent des actions autorisées par les règles de sécurité, sont spécifiées par des transitions.
- Les interdictions peuvent être implicites ou explicites. Elles sont implicites lorsqu'il n'y a pas de transition dans l'automate qui conduit à l'état souhaité (i.e. l'action demandée par le sujet). Les interdictions explicites, en revanche, sont spécifiées comme un "état d'échec" que le système générera si une action malveillante est détectée.
- Les obligations sont considérées comme des actions qui "doivent" être exécutées. Tout comme les autorisations, les obligations seront spécifiées par des transitions. Pour remplir le caractère "obligatoire", chaque transition se voit attribuer un délai d'expiration qui est remis à zéro si l'action est effectuée avant son expiration ; sinon, si le délai expire, une exception est générée et pourrait entraîner des sanctions.

La Figure 5.9 représente un automate temporisé spécifiant le scénario de notre étude de cas.

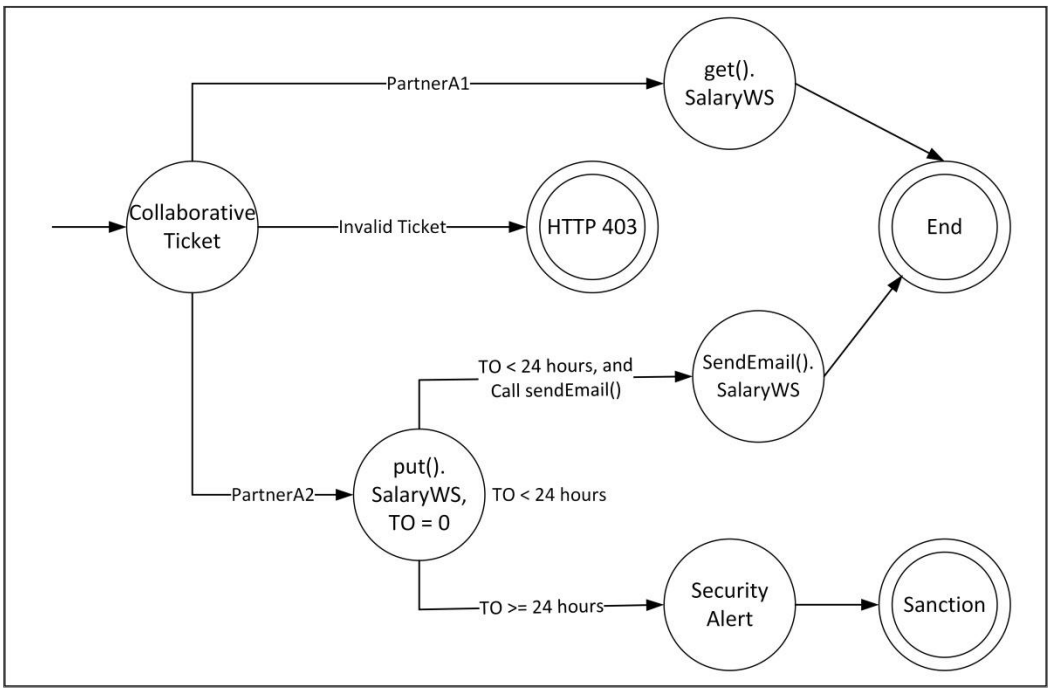

Figure 5.9. Exemple d'autorisation, d'interdiction et d'obligation avec un automate temporisé

Si Alice de l'organisation OrgA souhaite appeler la méthode "get", elle doit présenter un ticket chiffré par son organisation prouvant qu'elle est autorisée par OrgA à se présenter comme le sujet virtuel PartnerA1 de l'organisation OrgB. De plus, si elle souhaite faire appel à la méthode "put", elle doit présenter un autre ticket prouvant qu'elle est autorisée par OrgA à se présenter comme le sujet virtuel PartnerA2 de l'organisation OrgB. A la réception du ticket, le point d'application de la politique de l'organisation OrgB, le PEP, le déchiffre et vérifie sa validité. Si le ticket est valide, il le transmet au point de décision de la politique, le PDP, pour calculer la décision d'accès en fonction de la politique d'accès de l'organisation OrgB.

Pour intégrer les règles de sécurité, nous proposons d'étendre la spécification WS-Agreement et d'ajouter une nouvelle section, appelée "SecurityTerms", qui peut avoir la définition suivante :

```
<wsag:Terms>
…
<ac:SecurityTerms wsag:Name="xs:string" wsag:ServiceName="xs:string">
      <ac:Permissions wsag:Name="xs:string" wsag:ServiceName="xs:string">
            <ac:Transition>
                  <ac:Source>xs:string</ac:Source>
                  <ac:Target>xs:string</ac:Target>
                  <ac:Event>xs:string</ac:Event>
                  <ac:Action>xs:string</ac:Action>?
            </ac:Transition>+
      </ac:Permissions>?
      <ac:Obligations wsag:Name="xs:string" wsag:ServiceName="xs:string">
            <ac:Transition>
                  <ac:Source>xs:string</ac:Source>
                  <ac:Target>xs:string</ac:Target>
                  <ac:Event>xs:string</ac:Event>?
                  <ac:Guard>xs:string</ac:Guard>?
                  <ac:Action>xs:string</ac:Action>?
            </ac:Transition>+
      </ac:Obligations>?
      <ac:Prohibitions wsag:Name="xs:string" wsag:ServiceName="xs:string">
            <ac:Transition>
                  <ac:Source>xs:string</ac:Source>
                  <ac:Target>xs:string</ac:Target>
                  <ac:Event>xs:string</ac:Event>
            </ac:Transition>+
      </ac:Prohibitions>?
</ac:SecurityTerms >
…
</wsag:Terms>
```

Nous introduisons trois termes de sécurité : **Permissions**, **Obligations** et **Prohibitions**. Chacun de ces termes est constitué d'un ensemble de transitions. Nous considérons une transition comme étant un lien d'un état source vers un état cible lorsqu'un événement ou une garde se produit. Une transition peut éventuellement déclencher une action (par exemple, réinitialiser une horloge). De cette manière, l'automate de la Figure 5.9 peut être exprimé comme suit :

```
<wsag:Agreement xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement"
xmlns:ac="http://access-control.uca.ma">
…
<ac:SecurityTerms wsag:Name="PolyOrBAC Security Rules" wsag:ServiceName="SalaryWS">
      <ac:Permissions wsag:Name="PolyOrBAC Permissions" wsag:ServiceName="SalaryWS">
            <ac:Transition>
                  <ac:Source>Collaborative Ticket</ac:Source>
                  <ac:Event>PartnerA1</ac:Event>
                  <ac:Target>get</ac:Target>
            </ac:Transition>
            <ac:Transition>
                  <ac:Source>Collaborative Ticket</ac:Source>
                  <ac:Event>PartnerA2</ac:Event>
```

```
                    <ac:Target>put</ac:Target>
                    <ac:Action>TO = 0</ac:Action>
              </ac:Transition>
       </ac:Permissions>
       <ac:Obligations wsag:Name="PolyOrBAC Obligations" wsag:ServiceName="SalaryWS">
              <ac:Transition>
                    <ac:Source>put</ac:Source>
                    <ac:Guard>TO less than 24 hours</ac:Guard>
                    <ac:Target>put</ac:Target>
              </ac:Transition>
              <ac:Transition>
                    <ac:Source>put</ac:Source>
                    <ac:Guard>TO less than 24 hours</ac:Guard>
                    <ac:Event>call for sendEmail()</ac:Event>
                    <ac:Target>sendEmail()</ac:Target>
              </ac:Transition>
              <ac:Transition>
                    <ac:Source>put</ac:Source>
                    <ac:Guard>TO greater than or equal to 24 hours</ac:Guard>
                    <ac:Target>Security Alert</ac:Target>
              </ac:Transition>
       </ac:Obligations>
       <ac:Prohibitions wsag:Name="PolyOrBAC Prohibitions"
wsag:ServiceName="SalaryWS">
              <ac:Transition>
                    <ac:Source>Collaborative Ticket</ac:Source>
                    <ac:Event>Invalid Ticket</ac:Event>
                    <ac:Target>HTTP 403</ac:Target>
              </ac:Transition>
       </ac:Prohibitions>
</ac:SecurityTerms>
```

5.7. Contrôle d'accès dynamique pour les environnements collaboratifs

La Figure 5.10 illustre l'architecture du cadre de contrôle d'accès dynamique pour l'environnement collaboratif de notre étude de cas. Dans notre approche, l'agent consommateur (représentant l'organisation OrgA) et l'agent fournisseur (représentant l'organisation OrgB) négocient et créent des accords selon le protocole de la Figure 5.7. Une fois qu'un accord est créé et signé (les étapes 1 à 4), un utilisateur de l'organisation OrgA émet une demande d'accès, sous forme d'un ticket chiffré, au point d'application de la politique, le PEP, de l'organisation OrgB (étape 5). Ce dernier, après validation de l'identité de l'utilisateur et de l'agrément d'accès, transforme la demande en requête XACML qu'il transmet au point de décision de l'application, le PDP, (étape 6). Ce dernier calcule la décision en comparant la requête d'accès aux politiques de sécurité enregistrées dans le point de récupération de politique, le PRP (étape 7). Le PDP peut éventuellement se connecter à des sources externes d'attributs comme LDAP ou une base de données externe à travers le point d'information de la politique, le PIP (étape 8). Une fois que la décision est calculée, le PDP transmet la réponse au PEP (étape 9) pour l'appliquer (étape 10).

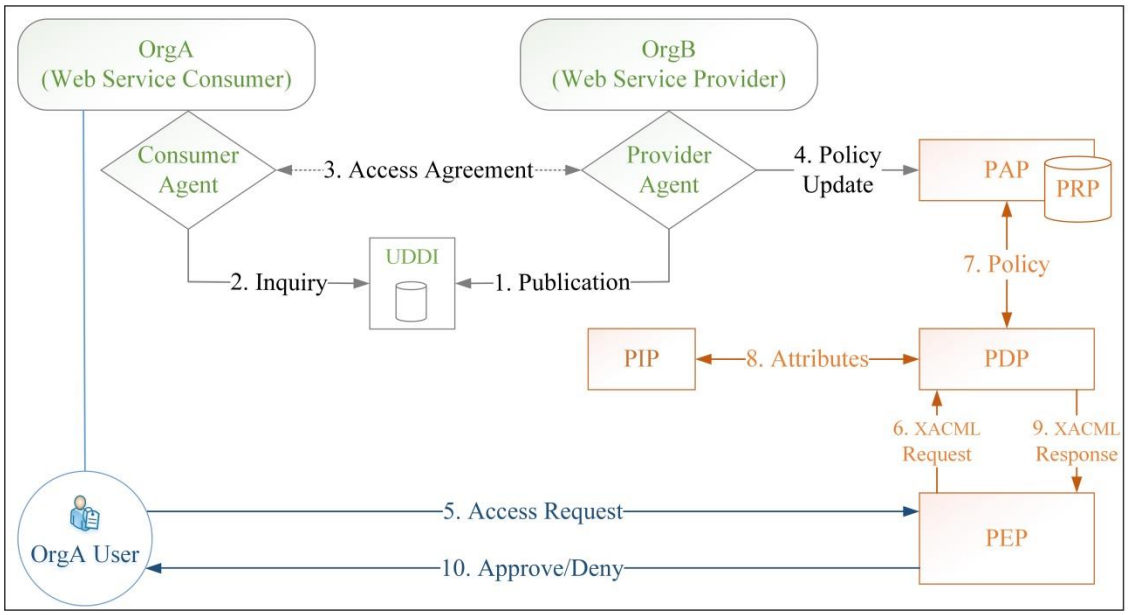

Figure 5.10. Cadre de contrôle d'accès collaboratif [Talha, 21-a]

Pour l'implémentation de notre solution, nous avons deux options :

- **WSAG4J** : Ce cadre a été publié dans sa dernière version 2.0 en 2012 et n'a pas été activement maintenu depuis (les certificats ont été expirés, il n'y a plus de support, trop faible communauté, etc.).
- **SLA-Framework** : ce cadre, plus récent, est un projet libre publié sous une licence Apache 2.0. Actuellement en version 1.1, SLA-Framework ne supporte pas les boucles de négociations (offres et contre-offres). Pour contourner cette limite, nous admettons que l'agent fournisseur ne peut qu'accepter ou refuser une offre faite par l'agent consommateur (à ce stade, nous nous dispensons de la possibilité des contre-offres).

Pour notre étude, nous avons opté pour SLA-Framework comme implémentation de la spécification WS-Agreement. C'est cette implémentation que nous avons étendue pour pouvoir exprimer des règles de sécurité PolyOrBAC. Plus précisément, nous avons adapté son code source pour prendre en charge la nouvelle section **SecurityTerms**. Pour ce faire, nous avons procédé comme suit :

- Cloner le projet à partir de son référentiel github : https://github.com/FIWARE/ops.Sla-framework.git.
- Installer tous les outils nécessaires au fonctionnement de l'application sous Windows (version 8.1), notamment: MySQL (version 5.5.45), Java (version 1.8.0), Maven (version 3.6.3), IntelliJ IDEA (version communautaire 2020.1) ainsi que Postman en tant que client REST.
- Créer une base de données vide (SLA-Framework s'occupe de la création de toutes les tables).
- Mettre à jour tous les fichiers de configuration pour les propriétés de la base de données, le chemin vers le serveur web Tomcat (une version par défaut de Tomcat est embarquée dans le projet SLA-Framework) ainsi que le chemin vers les journaux.
- Construire et tester le projet.

Comme le montre la Figure 5.11, six modules doivent être créés avec succès : SLA parent, SLA repository database, SLA Enforcement, SLA Tools, SLA Project Personalization et SLA Service.

```
work-master\sla-[INFO] Installing E:\BigData\SLAFramework\ops.Sla-framework-master\ops.Sla-framework-master\sla-core\sla-service\pom.xml
to C:\Users\Mohamed\[INFO] Installing E:\BigData\SLAFramework\ops.Sla-framework-master\ops.Sla-framework-master\sla-core\sla-service\pom.
xml to C:\Users\Mohamed\[INFO] Installing [INFO][INFO] Installing E:\BigData\SLAFramework\ops.Sla-framework-master\ops.Sla-framework-mast
er\sla-core\sla-service\pom.xml to C:\Users\Mo
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for SLA parent 0.1.1-SNAPSHOT:
[INFO]
[INFO] SLA parent ......................................... SUCCESS [  0.281 s]
[INFO] SLA repository database ............................ SUCCESS [  6.445 s]
[INFO] SLA Enforcement .................................... SUCCESS [  3.411 s]
[INFO] SLA Tools .......................................... SUCCESS [  0.896 s]
[INFO] SLA Project Personalization ........................ SUCCESS [  0.961 s]
[INFO] SLA Service ........................................ SUCCESS [  7.211 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  19.353 s
[INFO] Finished at: 2020-12-23T15:36:24+01:00
```

Figure 5.11. Construction du projet SLA-Framework

L'extension de SLA-Framework consiste à créer une nouvelle classe Java que nous avons appelée **SecurityTerm**. Cette nouvelle classe a été déclarée en tant que nouvel attribut de la classe eu.atos.sla.parser.data.wsag.AllTerms. La classe SecurityTerm définit trois nouveaux types : **Autorisations**, **Obligations** et **Interdictions**. Chacune de ces dernières est constituée d'une liste de Transitions. Une **Transition** définit les différents éléments nécessaires pour ajouter une règle de sécurité PolyOrBAC : **Source**, **Target**, **Event**, **Guard** et **Action**. Enfin, nous avons déclaré le nouvel espace de noms (**http://access-control.uca.ma**) dans la classe package-info.java. Nous avons ainsi étendu SLA-Framework afin qu'il puisse gérer les règles de sécurité PolyOrBAC. La Figure 5.12 montre un extrait d'un accord créé par cette nouvelle version de SLA-Framework.

POST http://localhost:8080/sla-service/agreements/

```
<ac:SecurityTerms wsag:Name="PolyOrBAC Security Rules" wsag:ServiceName="SalaryWS">
  <ac:Permissions wsag:Name="PolyOrBAC Permissions" wsag:ServiceName="SalaryWS">
      <ac:Transition>
          <ac:Source>Collaborative Ticket</ac:Source>
          <ac:Event>PartnerA1</ac:Event>
          <ac:Target>get().SalaryWS</ac:Target>
      </ac:Transition>
      <ac:Transition>
          <ac:Source>Collaborative Ticket</ac:Source>
          <ac:Event>PartnerA2</ac:Event>
          <ac:Target>put().SalaryWS</ac:Target>
          <ac:Action>TO = 0</ac:Action>
      </ac:Transition>
  </ac:Permissions>
  <ac:Obligations wsag:Name="PolyOrBAC Obligations" wsag:ServiceName="SalaryWS">
      <ac:Transition>
          <ac:Source>put().SalaryWS</ac:Source>
          <ac:Guard>TO less than 24 hours</ac:Guard>
          <ac:Target> put().SalaryWS </ac:Target>
      </ac:Transition>
      <ac:Transition>
          <ac:Source>put().SalaryWS</ac:Source>
          <ac:Guard>TO less than 24 hours</ac:Guard>
          <ac:Event>call for sendEmail().SalaryWS</ac:Event>
          <ac:Target>sendEmail().SalaryWS</ac:Target>
      </ac:Transition>
      <ac:Transition>
          <ac:Source>put().SalaryWS</ac:Source>
          <ac:Guard>TO greater than or equal to 24 hours</ac:Guard>
          <ac:Target>Security Alert</ac:Target>
      </ac:Transition>
  </ac:Obligations>
  <ac:Prohibitions wsag:Name="PolyOrBAC Prohibitions" wsag:ServiceName="SalaryWS">
      <ac:Transition>
          <ac:Source>Invalid Ticket</ac:Source>
          <ac:Event>Invalid Ticket</ac:Event>
          <ac:Target>HTTP 403</ac:Target>
      </ac:Transition>
  </ac:Prohibitions>
</ac:SecurityTerms>
</wsag:All>
</wsag:Terms>
</wsag:Agreement>
```

Figure 5.12. Exemple d'un accord PolyOrBAC généré par SLA-Framework étendu

Concernant l'environnement XACML, comme OrBAC utilise un langage formel basé sur la logique du premier ordre, le langage Prolog et l'outil SWI-Prolog répondent parfaitement à nos besoins. Concrètement, nous avons mis en place le PRP, le PAP, le PEP et le PDP comme expliqué ci-après.

Le PRP représente la base de connaissances. Pour notre scénario, cette base est initialisée avec les faits et les règles suivants, qui sont stockés dans un fichier appelé *polyrobac.pl* :

```
permission(orgb, director, salary_sheet, consult, financial_control).
permission(orgb, accountant, salary_sheet, update, salary_transfer).
employing(orgb, john, director).
employing(orgb, bob, accountant).
considering(orgb, get, consult).
considering(orgb, put, update).
using(orgb, salary_ws, salary_sheet).
defining(orgb, john, get, salary_ws, financial_control).
defining(orgb, bob, put, salary_ws, salary_transfer).

is_Permitted(X, get, salary_ws):-
permission(orgb, director, salary_sheet, consult, financial_control),
employing(orgb, X, director),
considering(orgb, get, consult),
using(orgb, salary_ws, salary_sheet),
defining(orgb, X, get, salary_ws, financial_control).

is_Permitted(Y, put, salary_ws):-
permission(orgb, accountant, salary_sheet, update, salary_transfer),
employing(orgb, Y, accountant),
considering(orgb, put, update),
using(orgb, salary_ws, salary_sheet),
defining(orgb, Y, put, salary_ws, salary_transfer).
```

Le PAP est représenté par un service web REST utilisé pour l'édition de la base de connaissances. Pour notre étude de cas, lorsqu'un accord est créé, signé et approuvé, l'agent fournisseur ajoute dans la base de connaissances les faits suivants :

```
employing(orgb, partner_a1, director).
employing(orgb, partner_a2, accountant).
defining(orgb, partner_a1, get, salary_ws, financial_control).
defining(orgb, partner_a2, put, salary_ws, salary_transfer).
```

Le PEP est représenté par un service web REST utilisé pour traiter les demandes d'accès des utilisateurs. A la réception d'une requête d'accès, sous forme d'un ticket de collaboration chiffré, ce module déchiffre le ticket et procède à une première étape de validation. Si le ticket est valide, il récupère le sujet virtuel, la méthode et le service web souhaités. Ces informations lui permettent ainsi de construire un prédicat qu'il transmet au PDP.

Le PDP est le moteur d'inférence qui vérifie si le prédicat reçu du PEP est vrai ou faux. Pour notre étude de cas, la Figure 5.13 montre les résultats de trois demandes faites par Alice pour :

- Appeler la méthode "put" du service web SalaryWS en tant que PartenerA1 (accès refusé).
- Appeler la méthode "get" du service web SalaryWS en tant que PartenerA2 (accès refusé).

- Appeler la méthode "put" du service web SalaryWS en tant que PartenerA2 (accès autorisé).

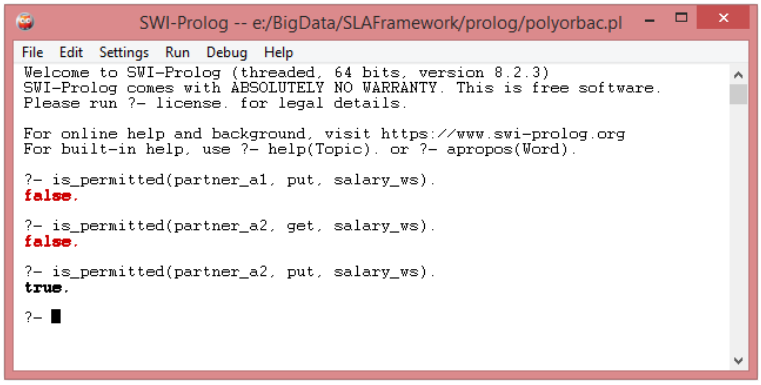

Figure 5.13. PolyOrBAC - Contrôle des interdictions et des autorisations

L'exécution de la méthode "put" sur le service web SalaryWS doit être enregistrée dans la base de faits avec l'horodatage d'exécution. Cela permettra de vérifier si Alice a rempli ses obligations. En effet, elle doit adresser, dans un délai de 24 heures maximum, un mail aux salariés concernés par le changement. La Figure 5.14 montre le contrôle des obligations.

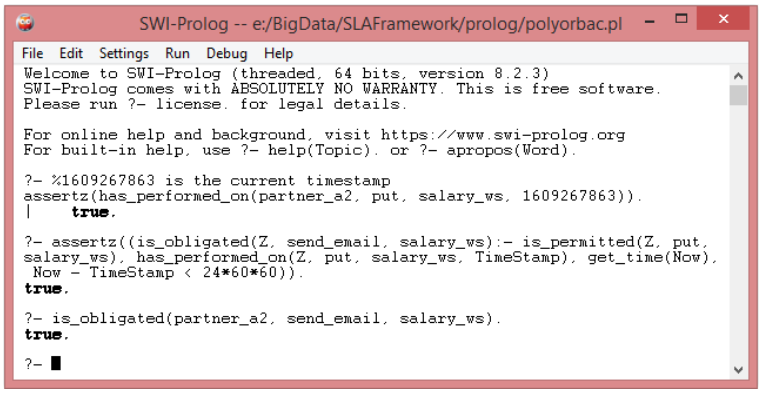

Figure 5.14. PolyOrBAC - Contrôle des obligations

5.8. Conclusion

Dans ce chapitre, nous avons proposé un cadre permettant de contrôler l'accès dans les environnements collaboratifs. C'est le résultat de l'extension et de la combinaison de deux cadres existant, à savoir PolyOrBAC et SLA-Framework. PolyOrBAC permet de contrôler l'accès dans les environnements collaboratifs mais souffre d'un problème majeur qui est la négociation manuelle des accords d'accès entre les acteurs d'une activité collaborative. SLA-Framework, qui est une implémentation du standard WS-Agreement, permet de contourner cette problématique mais ne supporte pas l'expression et la configuration des politiques et des règles de sécurité. L'extension de SLA-Framework pour couvrir ce besoin et son intégration dans PolyOrBAC ont permis d'automatiser le processus de négociation et de création des accords d'accès entre des agents représentant les fournisseurs et les consommateurs de données selon un protocole conforme au protocole de négociation normalisé de la spécification WS-Agreement.

La continuité de ce travail consiste à rendre PolyOrBAC indépendant du protocole SOAP (Simple Object Access Protocole) pour l’implémentation des services web. Aujourd’hui, les services web basés sur le modèle REST (REpresentational State Transfert) prennent de plus en plus de l’émergence au détriment de ceux basés sur le protocole SOAP. L’avantage de REST c’est qu’il est ouvert à différents formats d’échanges (JSON, HTML, Texte Plat et XML) tandis que SOAP est exclusivement basé sur XML qui est plus complexe à développer et plus lent dans les échanges (consommation des ressources, de la bande passante, etc.). Idéalement, PolyOrBAC devra supporter d’autres mécanismes de communication tels que l’accès distant, les notifications riches et pauvres, la communication par messages, etc.). En effet, dans les environnements Big Data et le Cloud Computing où l’on trouve différents locataires, il peut y avoir besoin d’accéder aux données hébergées dans le même lac de données par différents acteurs d’une activité collaborative ; le fait de passer par des services web, que ce soit SOAP ou REST, ne fera que compliquer le processus de communication. En ce qui concerne SLA-Framework, la version actuelle ne prend en charge que la négociation ponctuelle (elle ne supporte pas les boucles d’offres et de contre-offres) ; il faut donc penser à améliorer cette fonctionnalité ou développer notre propre implémentation de la spécification WS-Agreement. En plus de ces aspects techniques, la gestion explicite de la "confiance" va renforcer davantage la sécurité des plateformes collaboratives.

Chapitre 6. Vers un système de sécurité au service de la qualité de données en Big Data

6.1. Introduction

La qualité de données est un sujet essentiel pour tout projet Big Data. Les données de mauvaise qualité peuvent avoir des conséquences lourdes sur les projets et les entreprises. La sécurité de données est au même niveau d'importance ; si les données ne sont pas correctement protégées, les conséquences peuvent être néfastes. Comme nous l'avons expliqué dans [Talha, 19] et [Talha, 20-a], la mise en place d'un système de gestion de la qualité de données peut être bloquée par les mécanismes de sécurité. Cela peut conduire à des situations conflictuelles : faut-il autoriser certaines tolérances de sécurité pour simplifier la gestion de la qualité de données ou faut-il rester ferme d'un point de vue sécurité mais au détriment de la qualité de données ? Dans [Talha, 21-a], nous avons proposé un cadre de contrôle d'accès dynamique pour les plateformes collaboratives que l'on peut utiliser pour simplifier l'accès aux données tout en respectant les politiques de sécurité de chaque acteur. Dans ce chapitre, nous allons démontrer, à travers une étude de cas, qu'il est possible de déployer notre cadre de contrôle d'accès collaboratif dans une architecture qui permet d'éliminer les doublons dans les environnements distribués, de réduire la quantité de données et d'améliorer, en conséquence, leur qualité.

Un lac de données (ou Data Lake, en anglais) représente en général une plateforme réunissant plusieurs services, départements et organisations participant à son approvisionnement. La recherche en Big Data est toujours confrontée à de nombreuses problématiques complexes telles que l'intégration et la fusion des données hétérogènes, l'intégration et la fusion des politiques de contrôle d'accès, etc. Ces problèmes sont encore ouverts à la recherche et, à notre connaissance, aucune solution n'a encore été standardisée ou adoptée par une large communauté. Dans notre démarche, nous nous posons la question s'il y a des moyens permettant de contourner certains de ces problèmes. Il est vrai que nous disposons aujourd'hui des plateformes Big Data capables de stocker et de traiter rapidement de gros volumes de données hétérogènes telles que Hadoop et son écosystème ainsi que les bases de données NoSQL. Toutefois, il a été prouvé que ces plateformes souffrent de nombreux problèmes quant à la qualité et la sécurité de données. En outre, même si ces plateformes sont distribuées, que ce soit d'un point de vue stockage ou traitement de données, mais d'un point de vue architectural, on cherche au travers de ces plateformes à centraliser les données au même endroit qui est le lac de données. En revanche, si nous optons pour une architecture distribuée dans laquelle de nombreuses entités, chacune spécialisée dans un domaine, échangent des données au cours d'une activité collaborative, de nombreux problèmes de qualité et de sécurité seront systématiquement résolus. Les plateformes Big Data seront ainsi utilisées pour stocker les résultats des calculs effectués sur des entités distantes.

Notre approche consiste à traiter les environnements Big Data comme étant des plateformes collaboratives dans lesquelles les acteurs (ou les organisations) se font mutuellement confiance. Chaque acteur s'assure de la qualité des données qu'il gère et les met à disposition des autres acteurs en vertu d'un agrément d'accès. La suppression des doublons entre les différents acteurs de la plateforme collaborative permet ainsi d'éviter des processus complexes de gestion de la qualité de données tels que la synchronisation de données, la gestion de la cohérence et de la fraîcheur de données, etc. Pour cela, nous reprenons le cadre PolyOrBAC étendu [Talha, 21-a] et le déployons dans une architecture dédiée non seulement au contrôle d'accès mais aussi à la distribution de données.

Le reste de chapitre est organisé comme suit. La section 2 est une continuité du chapitre précédent ; nous y reprenons notre cadre de contrôle d'accès dynamique résultant de l'extension et de la combinaison

de PolyOrBAC et de SLA-Framework et l'intégrons dans un processus d'alimentation du Data Lake du processus d'évaluation de l'exactitude de données présenté dans le chapitre 4. Dans la section 3, nous présentons notre architecture permettant de nous servir d'un système de sécurité pour améliorer la qualité de données. Une étude de cas est ensuite présentée dans la section 4 pour démontrer l'intérêt de notre solution. La section 5 analyse et discute notre approche afin de montrer ses avantages et ses limites. Nous concluons enfin ce chapitre dans la section 6.

6.2. Evaluation sécurisée de l'exactitude de données en Big Data

Dans le cadre de nos travaux sur la qualité et la sécurité de données, nous avons proposé dans [Talha, 20-b] une démarche pour évaluer l'exactitude de données en Big Data. Nous avons également proposé dans [Talha, 21-a] un cadre, résultant de l'extension et de la combinaison de PolyOrBAC [Abou El Kalam, 07] et de SLA-Framework [SLA-Framework, 16], qui permet d'automatiser la négociation des accords d'accès et de contrôler dynamiquement l'accès aux plateformes collaboratives. Dans cette section, nous allons ressembler ces deux solutions pour obtenir un cadre sécurisé permettant l'évaluation de l'exactitude de données en Big Data.

Notre processus d'évaluation de l'exactitude de données est basé sur trois concepts :

- **Exactitude de données** : il s'agit d'une des principales dimensions de la qualité de données ; elle reflète le degré auquel les données d'un système d'information représentent le monde réel. Plus formellement, soit ν la valeur d'une donnée dans un système d'information et ν' la valeur de référence correspondante considérée comme correcte ; l'exactitude représente le degré de similarité entre ν et ν'.
- **Données de référence** : la mesure de l'exactitude de données doit se référer au monde réel qu'il modélise (par exemple, d'autres données que le système considère comme correctes, des connaissances humaines, etc.). L'obtention de ces "données de référence" est un processus très complexe. Notre approche d'identification des données de référence se base sur les critères d'exactitude.
- **Critères d'exactitude** : en pratique, les données à évaluer sont comparées à des données recueillies auprès d'une source de référence jugée suffisamment fiable. Les critères d'exactitude regroupent un ensemble de paramètres mesurables qui permettent d'évaluer la fiabilité des sources de données. Certains de ces critères correspondent aux fournisseurs de l'information (comme la confiance et la réputation) et d'autres correspondent aux données elles-mêmes (comme la cohérence, l'exhaustivité et l'actualité). Avant de lancer le processus d'évaluation de la qualité de données, chaque organisation, en fonction de ses besoins, doit déterminer les critères d'exactitude appropriés pour mesurer la fiabilité de ses sources d'informations.

L'approche d'évaluation de l'exactitude de données consiste en cinq étapes :

1. **Construction du Data Lake** : cette première étape consiste à collecter des données auprès des sources d'informations qui peuvent être internes ou externes à une organisation. Ces données sont ensuite stockées dans leur état brut pour alimenter le Data Lake.
2. **Construction de l'ensemble de données d'or (Golden Data Set)** : chaque ensemble de données du Data Lake se voit attribuer, sous forme de métadonnées, un niveau de fiabilité calculé en fonction de sa source d'origine et des critères d'exactitude prédéterminés par l'organisation. Ce

processus est ré-exécuté à chaque fois que des nouvelles données sont collectées ou lorsque les critères d'exactitude sont revus ou modifiés.

3. **Mise en correspondance de données** : la mise en correspondance des schémas des données à évaluer avec ceux des données hébergées dans le Golden Data Set est une étape nécessaire avant de procéder à l'identification des données de référence. Pour les données structurées, cela peut être réalisé grâce à des algorithmes de mise en correspondance de schémas. Pour les données semi-structurées et non-structurées, cette étape peut être plus complexe. Quelle que soit la nature des données, si les données d'entrée ne correspondent pas aux données du Golden Data Set, le processus d'évaluation se terminera avec un échec.
4. **Identification des données de référence** : les données de référence peuvent être construites grâce à un processus de couplage d'enregistrements appliqué aux données à évaluer et leurs correspondances dans le Golden Data Set. Concrètement, cette étape consiste à :
    - ignorer tous les ensembles de données, du Golden Data Set, qui ne répondent pas aux critères d'exactitude requis par l'organisation.
    - résoudre les potentiels conflits. Lorsque des mêmes informations existent dans différents ensembles de données et que tous ces derniers répondent aux critères d'exactitude, ce sont ceux les plus fiables qui sont considérés pour le reste du processus.
    - réduire le volume de données. Si la quantité de données est massive, il serait plus judicieux de se contenter d'un échantillon de données que l'on peut obtenir grâce à des algorithmes d'échantillonnage de données comme le "Reservoir Sampling" présenté dans le chapitre 4.
    - et, enfin, appliquer un algorithme de couplage d'enregistrements pour identifier les données de référence.
5. **Calcul de l'exactitude de données** : la dernière étape consiste à calculer la similarité entre les données à évaluer et celles de référence. Différentes méthodes existent pour mesurer la similarité entre des données telles que la distance de Levenshtein et la distance Jaro-Winkler pour les chaînes de caractères, la différence entre les valeurs numériques tout en considérant un seuil à partir duquel on peut considérer deux nombres comme étant similaires, les dates et les coordonnées géographiques peuvent être converties en nombres, etc.

La première étape de notre démarche consiste à collecter des données auprès de différentes organisations afin de pouvoir les exploiter dans le processus d'évaluation de l'exactitude de données. Cela ne peut être évident dans un environnement intelligent et volumineux. De nombreuses contraintes se posent telles que la recherche des fournisseurs de données, la négociation et la création des agréments d'accès, etc. Grâce à PolyOrBAC et à SLA-Framework étendus, présentés dans le chapitre 5, il est désormais possible d'automatiser les aspects de recherche et de collecte de données auprès des fournisseurs de données. La Figure 6.1 illustre notre nouvelle approche.

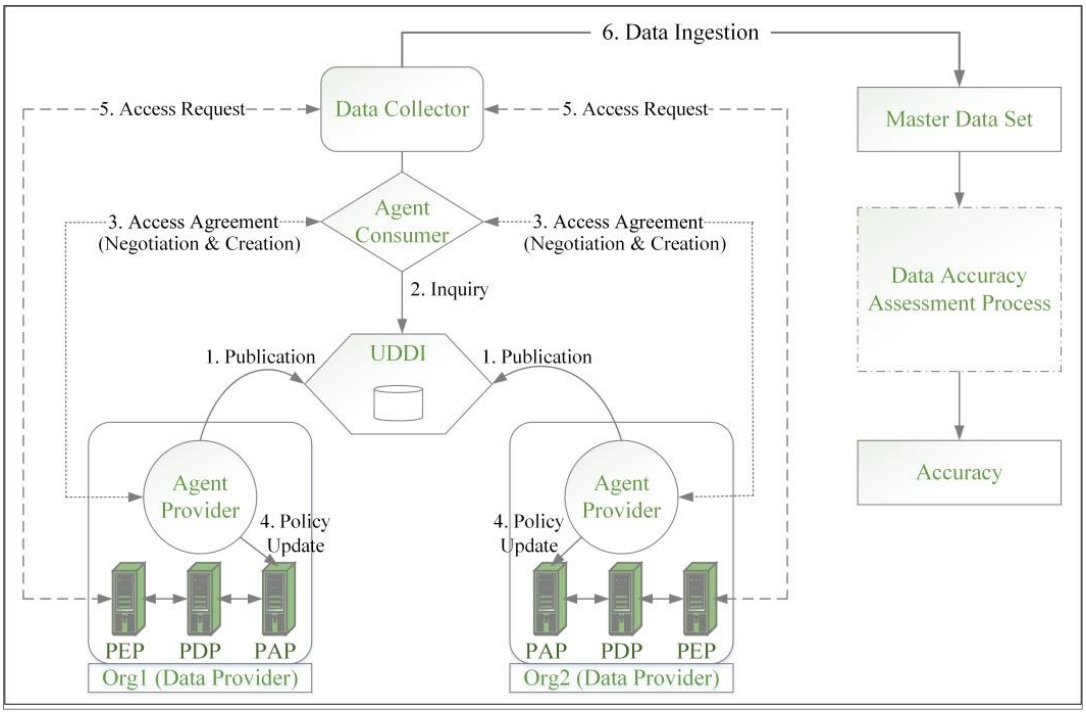

Figure 6.1. Cadre sécurisé pour l'évaluation de l'exactitude de données en Big Data [Talha, 21-a]

Notre nouvelle approche, pour alimenter en toute sécurité le Data Lake

dans le cadre d'une activité collaborative, consiste à utiliser PolyOrBAC et SLA-Framework étendus. Le principe consiste à ce que les fournisseurs de données (Org1, Org2, etc.) publient leurs services web dans le registre UDDI (Universal Description Discovery and Integration) de la plateforme collaborative via leurs agents (*étape 1. Publication*). Un collecteur de données dispose d'un agent consommateur qui surveille le registre UDDI afin d'identifier des services web qui peuvent l'intéresser (*étape 2. Inquiry*). Si un service web est identifié, l'agent consommateur contacte l'agent l'ayant publié pour négocier et créer un accord d'accès (*étape 3. Access Agreement*). A chaque fois qu'un nouvel accord est créé et signé, l'agent de publication met à jour la politique de sécurité de son organisation sur la base d'une architecture XACML (*étape 4. Policy Update*). Le collecteur de données peut enfin s'authentifier auprès des fournisseurs de données (*étape 5. Access Request*) pour accéder aux données et alimenter le Data Lake (*étape 6. Data Ingestion*).

6.3. La sécurité de données au service de la qualité de données

Les plateformes Big Data sont généralement conçues pour stocker les données dans des fichiers volumineux optimisés pour les opérations de lecture et d'écriture. Les opérations de modification, quant à elles, sont généralement lentes et coûteuses. Cela rend les processus de synchronisation des données entre différentes organisations complexes et coûteux. Les organisations préfèrent souvent conserver et gérer l'historique de toutes les valeurs de données pour éviter de les mettre à jour ou de les supprimer. En conséquence, la qualité des grands lacs de données se détériore avec le temps. Pour répondre à cette problématique, nous proposons une approche qui consiste à mettre en place une architecture distribuée basée sur un cadre de contrôle d'accès, dédié aux plateformes collaboratives, opérant comme distributeur

de données. Ce cadre devra également identifier et supprimer les doublons entre les différents acteurs de la plateforme collaborative. L'objectif est de réduire la quantité de données et d'améliorer leur qualité en évitant les processus de recopies de données qui impliquent des mécanismes de synchronisation et de gestion de la cohérence et de la fraîcheur de données. Étant dans un contexte Big Data, ce cadre doit répondre à certaines exigences telles que le grand volume, l'hétérogénéité et la grande vitesse de génération et de traitement de données. Entre autres, ce cadre doit être :

- **Évolutif** : la quantité de données transitant dans une plateforme collaborative augmente régulièrement, et le flux d'accès à contrôler augmente en conséquence. Ainsi, un cadre de contrôle d'accès doit avoir la capacité d'évoluer pour ne pas impacter le temps de traitement des demandes d'accès. De plus, étant donné que notre infrastructure agit comme une plateforme de distribution de données, l'ajout de nœuds pour gérer les appels des services web doit être automatique en fonction de la charge de la plateforme.
- **Dynamique** : le contrôle et la diffusion des données dans un environnement collaboratif sont généralement régis par des contrats (ou accords) d'accès établis entre les acteurs. Les termes d'un contrat peuvent évoluer dans le temps, de nouveaux contrats peuvent apparaître, d'autres peuvent disparaître, de nouveaux acteurs peuvent rejoindre la plateforme collaborative et d'autres peuvent la quitter. Ainsi, la gestion des contrats d'accès tout au long de leur cycle de vie (de la négociation à la création et jusqu'à la surveillance) doit être automatisée.
- **Indépendant des formats de données** : le cadre servant de distributeur de données doit supporter tout format de données (structurées, semi-structurées ou non-structurées).

La Figure 6.2 résume notre nouvelle approche. Nous fournissons une plateforme collaborative qui peut être déployée dans un environnement Big Data pour gérer le contrôle d'accès et la distribution de données entre différentes organisations. Notre cadre dispose d'un mécanisme pour identifier et supprimer les doublons entre ces organisations.

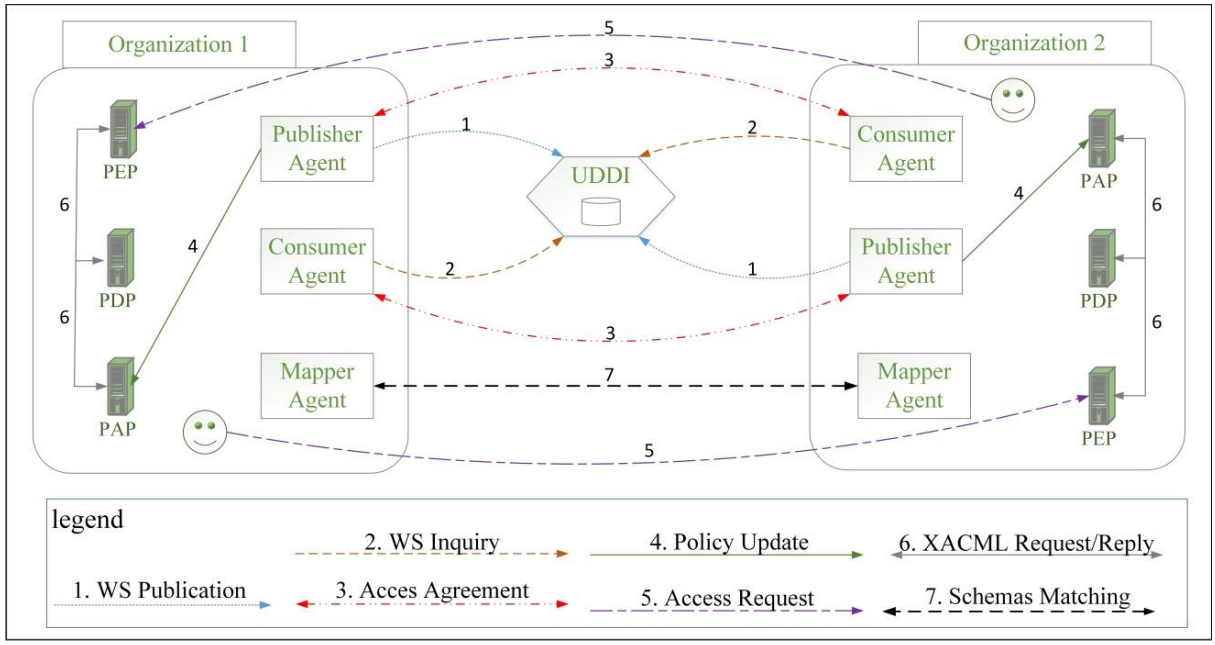

Figure 6.2. Cadre sécurisé de distribution de données dans un environnement collaboratif [Talha, 21-b]

Dans notre approche, chaque organisation de la plateforme collaborative doit disposer de quatre éléments :

- **Agent(s) de publication** : chaque organisation développe des services web de collaboration grâce auxquels d'autres organisations de la plateforme collaborative peuvent effectuer des actions sur ses données (GET, POST, PUT, PATCH et éventuellement DELETE). Ces services web sont publiés (*étape 1. WS Publication*) sur un registre public, l'UDDI, par des agents de publication.
- **Agent(s) consommateur(s)** : toute organisation dispose d'un ou plusieurs agents consommateurs qui supervisent l'UDDI (*étape 2. WS Inquiry*) pour identifier les services web qui peuvent être utilisés pour réaliser des activités collaboratives. Lorsqu'un service web est identifié, l'agent consommateur entre en phase de négociation pour mettre en place un accord d'accès avec l'agent l'ayant publié dans l'UDDI (*étape 3. Access Agreement*).
- **Composants XACML** : une fois que l'accord est créé, l'agent de publication met à jour la politique de sécurité de son organisation (*étape 4. Policy Update*) via le PAP pour ajouter les règles de sécurité permettant de réaliser l'activité collaborative. Lorsqu'un utilisateur souhaite appeler un service web (*étape 5. Access Request*), il soumet au PEP un ticket, généré et chiffré par son organisation, contenant des informations sur son identité ainsi que sur l'accord d'accès qu'il souhaite faire appliquer. Le PEP déchiffre le ticket, vérifie l'identité de l'émetteur et la validité de l'accord, et transmet la demande au PDP. Ce dernier calcule une décision d'accès qu'il renvoie au PEP (*étape 6. XACML Request/Reply*) pour l'appliquer.
- **Agent(s) de mise en correspondance** : lorsque deux organisations s'engagent dans une activité collaborative, il est recommandé de supprimer les doublons entre elles. Cela permettra de réduire la quantité de données que chaque organisation gère et d'améliorer la qualité des données transitant dans l'ensemble de la plateforme collaborative. Pour ce faire, les agents de mise en correspondance doivent, tout d'abord, identifier les données partagées, grâce notamment à des algorithmes de mise en correspondance de schémas et de couplage d'enregistrements, et ensuite déterminer les acteurs qui seront chargés de gérer chaque ensemble de données en commun (*étape 7. Schemas Matching*). Pour un jeu de données DSi, utilisé par deux organisations Org1 et Org2, trois scénarios sont possibles :
  - **Scénario 1** : parmi Org1 et Org2, un gestionnaire de DSi est identifié ; il sera le fournisseur de cet ensemble de données et devra mettre en place les mécanismes permettant au consommateur d'y accéder (services web, middleware, etc.).
  - **Scénario 2** : Org1 et Org2 veulent toutes les deux garder le contrôle sur DSi. Ce jeu de données restera donc dupliqué dans les deux organisations et chacune devra s'assurer de sa qualité (ce qui n'est pas optimal).
  - **Scénario 3** : Org1 et Org2, aucune d'elles ne veut prendre la responsabilité sur DSi. Dans ce cas, leurs agents consommateurs devront chercher dans la plateforme collaborative (dans l'UDDI) si une autre organisation Org3 utilise également DSi et peut se charger de gérer sa qualité et de le mettre à disposition des autres organisations (dans ce cas, c'est le scénario 1 qui sera appliqué). Si aucune organisation n'est trouvée, chaque organisation devra garder le contrôle sur DSi (dans ce cas, c'est le scénario 2 qui sera appliqué).

Notre cadre répond à toutes les exigences que nous avons déterminées pour se conformer aux prérequis des environnements Big Data :

- **Évolutivité** : le Cloud Computing apporte de nombreux avantages aux organisations pour mettre en place leurs infrastructures. Outre les coûts réduits, la simplicité de création et de surveillance des machines virtuelles, les infrastructures Cloud sont conçues pour être facilement évolutives, avec peu ou pas d'intervention humaine et sans interruption de service. Pour notre cadre, il est tout à fait possible de créer plusieurs instances virtuelles des agents de publication, de consommation et de mise en correspondance. Cela permettra de prendre en charge les énormes quantités de données transitant sur la plateforme tout en maintenant de bonnes performances.
- **Dynamicité** : des accords d'accès sont négociés entre les agents de publication (publiant les services web sur l'UDDI) et les agents de consommation intéressés par ces services web. De nombreuses solutions existent pour que les fournisseurs et les consommateurs de services web puissent négocier et créer des accords d'accès. Parmi ces solutions figurent WS-Agreement (Web Service Agreement) [Andrieux, 04] et son extension WS-Agreement Negotiation [Wäldrich, 11] qui sont les seules spécifications standardisées et adoptées par une large communauté [Marino, 18], [Li, 14-b]. Ce sont les seules solutions qui permettent de gérer des accords négociés bilatéralement. Ainsi, pour automatiser la négociation et la création des accords d'accès dans PolyOrBAC, nous proposons d'implémenter cette spécification en utilisant des cadres dédiés tels que WSAG4J ou SLA-Framework. Une implémentation de cette spécification a été détaillée dans le chapitre 5. De nombreux autres chercheurs ont utilisé WS-Agreement tels que Ludwig et al. [Ludwig, 06] qui ont été les premiers à l'appliquer pour négocier des contrats d'utilisation des ressources distribuées dans des environnements partagés. Smith et al. [Smith, 07] ont également utilisé cette spécification pour négocier des paramètres de sécurité tels que le niveau de cryptage et d'isolation des composants dans les grilles informatiques (Grid Computing). Li et al. [Li, 16] ont proposé une approche basée sur la spécification WS-Agreement permettant aux fournisseurs d'infrastructures en tant que service et à leurs consommateurs d'échanger des offres et des contre-offres jusqu'à ce qu'un accord soit trouvé concernant le niveau de service ainsi que les exigences de sécurité associées aux infrastructures. Récemment, dans [El Mokhtari, 21], les auteurs proposent une approche, basée sur les contrats intelligents et le WS-Agreement, permettant à des agents automatiques représentant les fournisseurs et les consommateurs de données de conclure des accords d'accès aux données qui gèrent les règles de sécurité et protègent la vie privée des individus.
- **Indépendance des formats de données** : la distribution de données dans notre cadre est principalement basée sur des services web. Les technologies de services web existantes aujourd'hui, que ce soit SOAP (Simple Object Access Protocol) ou REST (REpresentational State Transfer), peuvent prendre en charge tout type de données. Les fichiers XML des services web SOAP ou les fichiers JSON des services web REST peuvent contenir des données structurées, des données semi-structurées ou des références vers des fichiers lorsqu'il s'agit de transmettre des données non-structurées.

6.4. Etude de cas

Pour ce chapitre, nous allons reprendre l'étude de cas du chapitre 4 sur les gares ferroviaires de la région parisienne. Notre étude est basée sur des données ouvertes (open-data) téléchargées de cinq sources

de données : le site de la Région Ile-de-France (RIDF)[47], le site de la RATP[48], le site de la SNCF[49], le site de l'Île-de-France Mobilité (IDFM)[50] et le site du Gouvernement Français (GF)[51]. Un extrait des schémas de ces données ouvertes est représenté dans la Table 4.1 du chapitre 4. L'analyse de ces schémas a démontré deux constats :

- Les données de l'IDFM sont exactement les mêmes que celles de la RATP, et les données du GF sont exactement les mêmes que celles de la RIDF.
- Les données de la RATP, de la SNCF et de la RIDF sont différentes, n'ont pas la même structure et ont certaines colonnes en commun avec des libellés différents. Par exemple, certaines gares sont gérées à la fois par la SNCF et la RATP, mais chaque organisation gère des informations différentes.

Il est donc clair que ces cinq organisations peuvent construire une plateforme collaborative pour mieux gérer les données de transport à Paris et sa banlieue. L'IDFM et le GF peuvent s'appuyer respectivement sur les données de la RATP et de la RIDF. Cela permettra d'éviter les doublons et d'améliorer la qualité de données en évitant de nombreux processus de synchronisation et de gestion de cohérence. Cela peut être réalisé grâce à des accords d'accès aux données établis entre ces organisations et à un ensemble de mécanismes de notification et de distribution de données entre leurs plateformes. Par ailleurs, la RATP, la SNCF et la RIDF partagent certaines informations (comme les noms des gares et les coordonnées géographiques) et se spécialisent dans d'autres (comme le code INSEE et les équipements des annonces sonores et visuelles). Ces trois organisations doivent donc passer par un processus de mise en correspondance pour identifier qui doit gérer quels types d'informations.

Admettons que la RIDF est la seule organisation ayant le droit de modifier les noms des gares et les coordonnées géographiques (suite à des travaux d'élargissement par exemple) et que les codes des lignes ferroviaires sont gérés au niveau national par la SNCF. La mise en œuvre de notre modèle d'environnement collaboratif nécessite que ces cinq organisations aient quatre éléments :

- **Agent de publication** : la RIDF devra développer des services web permettant aux autres organisations de consulter les noms des gares et leurs coordonnées géographiques ainsi que des informations sur les équipements sonores et visuels. La SNCF devra développer des services web pour exposer les codes des lignes ferroviaires et des informations sur les villes et les départements des gares. Enfin, le code INSEE est géré par la RATP qui doit également rendre disponible cette information via un service web dédié. Le GF et l'IDFM, puisqu'ils s'appuient respectivement sur les données de la RIDF et de la RATP, n'ont pas besoin d'exposer de données. Tous les services web, avec leurs descriptions, doivent être publiés par les agents de publications de chaque organisation dans un registre public (par exemple, l'UDDI pour les services web SOAP).
- **Agent de consommation** : chaque organisation devra développer des agents consommateurs pour rechercher dans le registre public des services web leur permettant d'accéder aux données transitant dans la plateforme collaborative. Lorsqu'un agent consommateur identifie un service web qui l'intéresse, il initie un processus de négociation d'un accord d'accès avec l'agent l'ayant publié dans le registre public. Comme nous l'avons présenté dans la section précédente, ce

[47] https://data.iledefrance.fr/explore/dataset/gares-et-stations-du-reseau-ferre-dile-de-france-par-ligne/table/
[48] https://data.ratp.fr/explore/dataset/accessibilite-des-gares-et-stations-metro-et-rer-ratp/table/
[49] https://ressources.data.sncf.com/explore/dataset/liste-des-gares/table/
[50] https://data.iledefrance-mobilites.fr/explore/dataset/emplacement-des-gares-idf/table/
[51] https://www.data.gouv.fr/fr/datasets/accessibilite-des-gares-et-stations-de-metro-et-rer-ratp-1/

processus de gestion des accords d'accès peut être automatisé grâce à la spécification WS-Agreement comme nous l'avons détaillé dans [Talha, 21-a].

- **Composants XACML** : à la fin du processus de négociation entre un agent de publication AP_x d'une organisation ORG_x et un agent consommateur AC_y d'une organisation ORG_y, l'agent AP_x doit mettre à jour la politique de sécurité de l'organisation ORG_x, via son PAP, pour permettre à l'agent AC_y et aux utilisateurs de l'organisation ORG_y d'utiliser les services web dans le cadre d'un accord d'accès. Le demandeur d'accès peut alors contacter le PEP de l'organisation ORG_x pour formuler des requêtes d'accès en présentant un ticket chiffré d'accès collaboratif contenant son identité, la référence de l'accord d'accès qu'il souhaite appliquer et le service web qu'il souhaite appeler.
- **Agent de mise en correspondance** : le GF et l'IDFM n'ont pas besoin d'agents de mise en correspondance puisqu'ils récupèrent toutes leurs données auprès de la RIDF et de la RATP. Concernant la RATP, la SNCF et la RIDF, chacune d'entre elles gère certaines informations (ex. la RIDF gère les noms des gares et leurs coordonnées géographiques, la SNCF gère les informations sur les codes des lignes ferroviaires, etc.). Avant de démarrer l'activité collaborative, ces informations sont dupliquées entre certaines organisations. Cela peut créer des problèmes de qualité de données si les processus de synchronisation ne sont pas fiables. Les agents de mise en correspondance devront donc identifier les jeux de données dupliqués et déterminer les organisations de "référence" qui se chargeront d'assurer la qualité de ces jeux de données et de les mettre à disposition des autres organisations. Les organisations de référence devront également informer les consommateurs de tout changement de données (ajout, modification ou suppression).

6.5. Analyse et Discussion

La donnée est aujourd'hui un élément clé autour duquel s'articulent de nombreuses activités économiques et sociales. La qualité et la sécurité de données sont deux processus essentiels pour tout projet centré sur les données. Les environnements Big Data représentent un ensemble de nouvelles technologies, modèles et techniques de collecte, de stockage, d'analyse et d'extraction de valeurs à partir de très gros volumes de données. La tendance aujourd'hui est de centraliser les données dans un lac unique dans lequel toutes les structures d'une organisation stockent leurs données. Cela présente de nombreux avantages pour les entreprises (haute disponibilité, coûts d'hébergement réduits, IT verte, etc.) mais pose, en revanche, de nombreux défis à la qualité et à la sécurité de données.

Dans nos travaux de recherche, nous examinons les conflits qui peuvent exister entre les systèmes de gestion de la qualité de données et les systèmes de sécurité de données dans le contexte Big Data. La gestion de la qualité se fait généralement en deux phases : « l'évaluation », pour identifier les lacunes de qualité, et ensuite, « l'amélioration », pour corriger ces lacunes. L'évaluation de la qualité de données peut nécessiter un accès en lecture aux données collectées auprès de différentes organisations dans une plateforme collaborative (pour pouvoir effectuer des opérations de comparaison). L'amélioration de la qualité de données, de sa part, peut nécessiter un accès en écriture (pour mettre à jour les données obsolètes, supprimer les doublons, compléter les données manquantes, etc.). Ces deux processus peuvent être bloqués par les politiques et les mécanismes de sécurité, ce qui peut créer des potentiels conflits auprès des organisations. Dans ce chapitre, nous avons prouvé qu'il est possible de se servir des systèmes de sécurité pour améliorer la qualité de données. Notre cadre, résultant de l'extension et de l'intégration de

PolyOrBAC et de SLA-Framework, a été déployé en tant que système de contrôle d'accès et de distribution de données pour les plateformes collaboratives. Cela apporte de nombreux avantages tels que la gestion des politiques de sécurité des données hétérogènes et la suppression des données dupliquées.

6.6. Conclusion

La gestion de la qualité de données et la sécurité de données sont deux systèmes essentiels à l'exploitation efficace et efficiente de données, en particulier dans le contexte de la Big Data. Afin d'éviter des conflits mutuels entre ces deux systèmes, il est souvent nécessaire de procéder à une rationalisation minutieuse. Dans ce chapitre, nous avons rappelé que l'accès en lecture et/ou en écriture aux données d'une organisation, pour gérer leur qualité, peut être bloqué par les politiques de sécurité de cette organisation elle-même. En effet, les grandes structures sont souvent organisées en sous-organisations autonomes, parfois indépendantes, gérant chacune des domaines d'activité et des jeux de données privés. Considérer la gestion de la qualité de données comme une activité collaborative à laquelle participent toutes les structures d'une organisation semble résoudre une partie de ce problème. De nombreux modèles ont été proposés dans la littérature pour sécuriser les plateformes collaboratives mais, à notre connaissance, tous ont des limites en ce qui concerne les contraintes Big Data (Volume, Variété, Vélocité, etc.). Nous avons opté dans ce chapitre d'utiliser un cadre résultant de l'extension de PolyOrBAC et de SLA-Framework comme étant un système de contrôle d'accès et, en même-temps, un système de distribution de données dans les plateformes collaboratives. L'objectif est de minimiser l'intervention humaine et d'automatiser l'activité collaborative grâce à un système combinant le contrôle d'accès et la distribution de données. Ce système de sécurité permet d'optimiser le stockage de données en supprimant les données dupliquées et, par conséquent, d'améliorer la qualité des données transitant sur les plateformes collaboratives.

Pour faire avancer nos travaux de recherche, il sera nécessaire de travailler sur l'aspect sécurité soit en améliorant encore le cadre PolyOrBAC soit en mettant en place une autre solution dédiée à la sécurisation des activités collaboratives sur les environnements Big Data. En outre, l'hétérogénéité des données devra être davantage abordée dans le processus d'évaluation de la qualité de données, en particulier pour les données non-structurées. En effet, les statistiques montrent que près de 95% des données produites aujourd'hui sont non-structurées [Adnan, 19], [Seng, 19]. La recherche des correspondances dans le lac de données et le calcul des similarités sont des processus très complexes pour les données non-structurées. Cela peut nécessiter éventuellement une étape de classification de données et des mécanismes adaptés à chaque format de données pour gérer leur qualité.

Conclusion Générale

Ce travail de thèse étudie la qualité et la sécurité de données en Big Data. Il traite particulièrement le conflit qui peut exister entre les processus d'évaluation de la qualité de données, d'une part, et les mécanismes de sécurité de données, d'autre part. La qualité de données peut être évaluée à travers des dimensions mesurables comme l'exactitude, la complétude, la cohérence et la fraîcheur de données. Pour ce travail, nous nous sommes contentés de l'étude de l'exactitude, la dimension la plus importante et la plus complexe à mesurer. Pour ce faire, nous avons proposé un cadre d'évaluation de l'exactitude de données en formalisant le concept des "critères d'exactitude" qui permet d'identifier les "données de référence" considérées correctes par le système. Pour répondre aux exigences de la Big Data (volume important de données, hétérogénéité de données, grande vitesse de traitement, etc.), l'intégration de l'échantillonnage de données grâce à l'algorithme "Reservoir Sampling" permet de réduire le volume de données, de gagner en efficacité et d'optimiser le temps de traitement. Cet algorithme est parfaitement applicable aux flux de données pour le traitement à la volée. Concernant l'hétérogénéité, nous nous contentons dans ce travail des données structurées auxquelles des algorithmes de mise en correspondance et de couplage d'enregistrements ont été appliqués. Les données semi-structurées et non-structurées, quant à elles, nécessitent des traitements supplémentaires que nous aborderons dans un travail futur. En conclusion, l'évaluation de la qualité de données, de manière générale, nécessite des comparaisons avec d'autres données qui peuvent être protégées par différents mécanismes et politiques de sécurité. Etant donné que l'intégration des politiques de sécurité hétérogènes constitue un grand problème de recherche en Big Data, nous avons proposé une approche pour contourner ce problème. Notre solution consiste à instaurer un environnement regroupant plusieurs acteurs, chacun garde son propre système de sécurité, autour d'une activité collaborative qui est "la gestion de la qualité de données". Pour ce faire, nous avons étendu et intégré deux cadres existants : PolyOrBAC et SLA-Framework. PolyOrBAC est destiné au contrôle d'accès dans les plateformes collaboratives mais présente une limite vis-à-vis des environnements Big Data : la négociation des agréments d'accès aux données est un processus manuel mené en amont entre les acteurs impliqués dans l'activité collaborative. Pour résoudre ce problème, nous avons automatisé tout le processus de gestion des agréments d'accès grâce à la spécification WS-Agreement à travers son implémentation SLA-Framework. Pour cela, nous avons proposé un nouveau procédé de négociation PolyOrBAC conforme au protocole de négociation WS-Agreement. Nous avons ensuite étendu SLA-Framework pour pouvoir supporter les règles de sécurité PolyOrBAC. L'intégration de ces deux cadres a permis d'obtenir un nouveau cadre de contrôle d'accès dynamique pour les environnements collaboratifs. Dans ce nouveau cadre, la négociation des agréments d'accès et des regèles de sécurité ainsi que la gestion de ces agréments (création, re-négociation et surveillance) sont confiées à des agents automatiques, représentant les fournisseurs et les consommateurs de données. Par ailleurs, l'utilisation de notre cadre comme étant une plateforme de distribution de données, permet de supprimer les données dupliquées chez les acteurs participant à l'activité collaborative et d'améliorer la qualité de données transitant sur tout l'environnement collaboratif.

La continuité de ce travail devra aborder les données semi-structurées et non-structurées. Celles-ci sont générées en grandes quantités par les objets connectés, les réseaux sociaux et les algorithmes de l'intelligence artificielle. En plus, les bases de données NoSQL sont de plus en plus adoptées pour les grands projets au même niveau que le stockage sur le Cloud. Il est donc primordial de s'intéresser aux données semi-structurées et non-structurées. Concernant le cadre PolyOrBAC, nous devons le rendre

indépendant du protocole SOAP, voir indépendant des technologies des services web. Le mode de communication dans PolyOrBAC doit être abstrait afin de pouvoir y intégrer les technologies adaptées pour chaque projet. En ce qui concerne SLA-Framework, bien qu'une nouvelle version prenant en considération les boucles de négociation (offres et contre-offres) soit attendue, ce sera mieux d'intégrer notre extension dans le projet SLA-Framework ou développer notre propre implémentation de la spécification WS-Agreement. Enfin, la gestion de la "confiance" renforcera davantage la sécurité de données dans les plateformes collaboratives ; il s'agit d'un concept que nous ne traitons pas explicitement et sur lequel nous devons porter plus d'attention.

Bibliographie

[Abou El Kalam, 03-a] Abou El Kalam, Anas & Baida, R.E. & Balbiani, P. & Benferhat, S. & Cuppens, Frédéric & Deswarte, Yves & Miege, A. & Saurel, Claire & Trouessin, G. (2003). Organization based access control. 120 - 131. DOI: 10.1109/POLICY.2003.1206966.

[Abou El Kalam, 03-b] Abou El Kalam, Anas. (2003). Modèles et politiques de securite pour les domaines de la santé et des affaires sociales. Réseaux et télécommunications. Institut National Polytechnique de Toulouse - INPT.

[Abou El Kalam, 07] Abou El Kalam, Anas & Deswarte, Yves & Baina, Amine & Kaaniche, Mohamed. (2007). Access Control for Collaborative Systems: A Web Services Based Approach. 1064-1071. 10.1109/ICWS.2007.30.

[Abou El Kalam, 09] Abou El Kalam, Anas & Deswarte, Yves & Baina, Amine & Kaaniche, Mohamed. (2009). PolyOrBAC: A security framework for Critical Infrastructures. International Journal of Critical Infrastructure Protection. 2. 154-169. DOI: 10.1016/j.ijcip.2009.08.005.

[Adnan, 19] Adnan, Kiran & Akbar, Rehan. (2019). An analytical study of information extraction from unstructured and multidimensional big data. Journal of Big Data. 6. DOI: 10.1186/s40537-019-0254-8.

[Ahn, 00] Ahn, Gail-Joon & Sandhu, Ravi. (2000). Role-Based Authorization Constraints Specification. ACM Trans. Inf. Syst. Secur.. 3. 207-226. DOI: 10.1145/382912.382913.

[Ait Aali, 15] Ait Aali, Nawal & Baina, Amine & Echabbi, Loubna. (2015). Trust integration in Collaborative Access Control model for Critical Infrastructures. DOI: 10.1109/SITA.2015.7358427.

[Alladi, 18] Alladi, Bhima Sankaram & Prasad, Srinivas. (2017). Big data life cycle: security issues, challenges, threat and security model. International Journal of Engineering & Technology. Volume 7. DOI: 10.14419/ijet.v7i1.3.9666.

[Andrieux, 04] Andrieux, Alain & Czajkowski, Karl & Dan, Asit & Keahey, Kate & Ludwig, Heiko & Nakata, Toshiyuki & Pruyne, Jim & Rofrano, John & Tuecke, Steve & Xu, Ming. (2004). Web services agreement specification (WS-Agreement). Global Grid Forum. 2.

[Arolfoand, 18] Arolfo, Franco & Vaisman, Alejandro. (2018). Data Quality in a Big Data Context. DOI: 10.1007/978-3-319-98398-1_11.

[Aumueller, 05] Aumueller, David & Do, Hong & Massmann, Sabine & Rahm, Erhard. (2005). Schema and ontology matching with COMA++. Proceedings of the ACM SIGMOD International Conference on Management of Data. 906-908. DOI: 10.1145/1066157.1066283.

[Azaria, 16] Azaria, Asaph & Ekblaw, Ariel & Vieira, Thiago & Lippman, Andrew. (2016). MedRec: Using Blockchain for Medical Data Access and Permission Management. 25-30. DOI: 10.1109/OBD.2016.11.

[Azmi, 15] Azmi, Zal. (2015). Opportunities and Security Challenges of Big Data. Current and Emerging Trends in Cyber Operations. DOI: 10.1057/9781137455550_12.

[Back, 02] Back, Adam. (2002). Hashcash - A Denial of Service Counter-Measure.

[Back, 97] Back, Adam. (1997). Hashcash. Available online: http://www.hashcash.org/ (last accessed December 25, 2021)

[Baig, 17] Baig, Ramsha & Khan, Waqas & Haq, Irfan & Khan, Irfan. (2017). Agent-Based SLA Negotiation Protocol for Cloud Computing. 33-37. DOI: 10.1109/ICCCRI.2017.13.

[Ballou, 85] Ballou, Donald & Pazer, Harold. (1985). Modeling Data and Process Quality in Multi-Input, Multi-Output Information Systems. Management Science. 31. 150-162. DOI: 10.1287/mnsc.31.2.150.

[Bao, 18] Bao, Rongxin & Chen, Zhikui & Obaidat, Mohammad. (2018). Challenges and techniques in Big data security and privacy: A review. Security and Privacy. 1. e13. DOI: 10.1002/spy2.13.

[Batini, 06] Batini, Carlo & Scannapieco, Monica. (2006). Data Quality: Concepts, Methodologies and Techniques. DOI: 10.1007/3-540-33173-5.

[Batini, 09] Batini, Carlo & Cappiello, Cinzia & Francalanci, Chiara & Maurino, Andrea. (2009). Methodologies for Data Quality Assessment and Improvement. ACM Comput. DOI: 10.1145/1541880.1541883.

[Batini, 15] Batini, Carlo & Rula, Anisa & Scannapieco, Monica & Viscusi, Gianluigi. (2015). From Data Quality to Big Data Quality. Journal of Database Management. DOI: 10.4018/978-1-4666-9840-6.ch089.

[Battre, 10] Battré, Dominic & Brazier, Frances & Clark, Kas & Oey, Michel & Papaspyrou, Alexander & Wäldrich, Oliver & Wieder, Philipp & Ziegler, Wolfgang. (2010). A Proposal for WS-Agreement Negotiation. 233-241. DOI: 10.1109/GRID.2010.5697976.

[Becker, 09] Becker, Jörg & Poeppelbuss, Jens & Gloerfeld, Fabian & Bruhns, Peter. (2009). The Impact of Data Quality on Value Based Management of Financial Institutions. 490.

[Belbergui, 16] Belbergui, Chaimaa & EL KAMOUN, Najib & Rachid, Hilal. (2016). A Dynamic Access Control Model for Cloud Computing Environments. 21-29. DOI: 10.1145/3017971.3017979.

[Bell, 76] Bell, D.E. & La Padula, L.J. (1976). Secure Computer System: Unified Exposition and Multics Interpretation. Deputy For Command And Management Systems. Electronic Systems Division, Air Force Systems Command, United States Air Force. ESD-TR-75-306. MTR-2997 Rev.1.

[Bellahsene, 11] Bellahsene, Zohra & Bonifati, Angela & Rahm, Erhard. (2011). Schema Matching and Mapping. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-642-16517-7. DOI: 10.1007/978-3-642-16518-4.

[Ben Saidi, 12] Ben Saidi, Mustapha & Abou El Kalam, Anas & Marzouk, Abderrahim. (2012). TOrBAC: A Trust Organization Based Access Control Model for Cloud Computing Systems. International Journal of Soft Computing and Engineering (IJSCE). ISSN: 2231-2307, Volume-2 Issue-4.

[Ben Saidi, 13] Ben Saidi, Mustapha & Marzouk, Abderrahim. (2013). Multi-Trust_OrBAC: Access Control Model for Multi-Organizational Critical Systems Migrated To the Cloud. 2231-2307.

[Benhadj Djilali, 19] Benhadj Djilali, Hadjer & Tandjaoui, Djamel. (2019). Dynamic Team Access Control for Collaborative Internet of Things: 4th International Conference, MSPN 2018, Paris, France, June 18-20, 2018, Revised Selected Papers. DOI: 10.1007/978-3-030-03101-5_7.

[Bernstein, 11] Bernstein, Philip & Madhavan, Jayant & Rahm, Erhard. (2011). Generic Schema Matching, Ten Years Later. PVLDB. 4. 695-701. DOI: 10.14778/3402707.3402710.

[Bertino, 15] Bertino, Elisa. (2015). Big Data - Security and Privacy. IEEE International Congress on Big Data, DOI: 10.1109/BigDataCongress.2015.126.

[Bertoni, 08] Bertoni, Guido & Daemen, Joan & Peeters, Michaël & Van Assche, Gilles. (2008). Keccak speciecations. Available online: https://keccak.team/index.html (last accessed December 25, 2021)

[Bethencourt, 07] Bethencourt, John & Sahai, Amit & Waters, Brent. (2007). Ciphertext-Policy Attribute-Based Encryption. Proceedings - IEEE Symposium on Security and Privacy. 321-334. DOI: 10.1109/SP.2007.11.

[Biba, 77] Biba, Ken. (1977). Integrity Considerations for Secure Computer Systems. Deputy For Command And Management Systems. Electronic Systems Division, Air Force Systems Command, United States Air Force. ESD-TR-76-372. MTR-3153 Rev.1.

[Bovee, 03] Bovee, Matthew & Srivastava, Rajendra & Mak, Brenda. (2003). A Conceptual Framework and Belief-Function Approach to Assessing Overall Information Quality. Int. J. Intell. Syst.. 18. 51-74. DOI: 10.1002/int.10074.

[Cafarella, 08] Cafarella, Michael & Halevy, Alon & Wang, Daisy & Wu, Eugene & Zhang, Yang. (2008). WebTables: Exploring the power of tables on the web. PVLDB. 1. 538-549. DOI: 10.14778/1453856.1453916.

[Cai, 15] Cai, Li & Zhu, Yangyong. (2015). The Challenges of Data Quality and Data Quality Assessment in the Big Data Era. Data Science Journal. 14. DOI: 10.5334/dsj-2015-002.

[Cai, 18] Cai, Fangbo & Zhu, Nafei & He, Jingsha & Mu, Pengyu & Li, Wenxin & Yu, Yi. (2018). Survey of access control models and technologies for cloud computing. Cluster Computing. 22. DOI: 10.1007/s10586-018-1850-7.

[Cantor, 05] Cantor, Scott & Moreh, Jahan & Philpott, Rob & Maler, Eve. (2005). Metadata for the OASIS Security Assertion Markup Language (SAML). Version 2.0. Available online: https://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf (last accessed December 25, 2021)

[Cappiello, 04] Cappiello, Cinzia & Francalanci, Chiara & Pernici, Barbara. (2004). Data quality assessment from the user's perspective. ACM 1-58113-902-0/04/0006. 68-73. DOI: 10.1145/1012453.1012465.

[Castro, 15] Castro, Ignacio & Panda, Aurojit & Raghavan, Barath & Shenker, Scott & Gorinsky, Sergey. (2015). Route Bazaar: Automatic Interdomain Contract Negotiation. The 15th Workshop on Hot Topics in Operating Systems (HotOS XV 2015). 18-20 May 2015.

[Cavoukian, 14] Cavoukian, Ann & Castro, Daniel. (2014). Big Data and Innovation , Setting the Record Straight : De-identification Does Work. Information and Privacy Commissioner. 1-18.

[Chang, 15] Chang, Victor. (2015). Towards a Big Data System Disaster Recovery in a Private Cloud. Ad Hoc Networks. 35. DOI: 10.1016/j.adhoc.2015.07.012.

[Che, 13] Che, Dunren & Safran, Mejdl & Peng, Zhiyong. (2013). From Big Data to Big Data Mining: Challenges, Issues, and Opportunities. Proc Of DASFAA-BDMA Workshop. 7827. 1-15. DOI: 10.1007/978-3-642-40270-8_1.

[Chen, 14-a] Chen, C.L. Philip & Zhang, Chun-Yang. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information Sciences. 275. 314–347. DOI: 10.1016/j.ins.2014.01.015.

[Chen, 14-b] Chen, Hongsong & Bhargava, Bharat & Zhongchuan, Fu. (2014). Multilabels-Based Scalable Access Control for Big Data Applications. IEEE Cloud Computing. 1. 65-71. DOI: 10.1109/MCC.2014.62.

[CNIL, 16] Commission Nationale de l'Informatique et des Libertés (CNIL). (2016). Comprendre les grands principes de la cryptologie et du chiffrement. Available online: https://www.cnil.fr/fr/comprendre-les-grands-principes-de-la-cryptologie-et-du-chiffrement (last accessed December 25, 2021)

[CNIL, 18-a] Commission Nationale de l'Informatique et des Libertés (CNIL). (2018). Le règlement général sur la protection des données - RGPD. Available online: https://www.cnil.fr/fr/reglement-europeen-protection-donnees (last accessed December 25, 2021)

[CNIL, 18-b] Commission Nationale de l'Informatique et des Libertés (CNIL). (2018). Authentification par mot de passe : les mesures de sécurité élémentaires. Available online: https://www.cnil.fr/fr/mot-de-passe (last accessed December 25, 2021)

[Cole, 05] Cole, Gary. (2005). OASIS Service Provisioning Markup Language (SPML). Version 2. Available online: https://www.oasis-open.org/standard/spml/ (last accessed December 25, 2021)

[Copas, 90] Copas, J & Hilton, F. (1990). Record Linkage: Statistical Models for Matching Computer Records. Journal of the Royal Statistical Society. Series A, (Statistics in Society). 153. 287-320. DOI: 10.2307/2982975.

[Cortez, 13] Cortez, Eli & Da Silva, Altigran. S.. (2013). Unsupervised Information Extraction by Text Segmentation. Springer briefs in computer science. ISBN: 978-3-319-02596-4. DOI: 10.1007/978-3-319-02597-1.

[Coutinho, 14] Coutinho, Carlos & Cretan, Adina & Ferreira da Silva, Catarina & Ghodous, P. & Jardim-Goncalves, Ricardo. (2014). Service-based Negotiation for Advanced Collaboration in Enterprise Networks. Journal of Intelligent Manufacturing. 27. DOI: 10.1007/s10845-013-0857-4.

[Crosby, 79] Crosby Phil. (1979). Quality is free. New York:McGraw-Hill.

[CSA, 16] Cloud Security Alliance (CSA). (2016). Big Data Working Group. Big Data Security and Privacy Handbook - 100 Best Practices in Big Data Security and Privacy.

[cybernews, 21] cybernews. (2021). COMB: largest breach of all time leaked online with 3.2 billion records. Available online: https://cybernews.com/news/largest-compilation-of-emails-and-passwords-leaked-free/ (last accessed December 25, 2021)

[Damen, 14] Damen, Stan & Hartog, Jerry & Zannone, Nicola. (2014). CollAC: Collaborative access control. 2014 International Conference on Collaboration Technologies and Systems, CTS 2014. 142-149. DOI: 10.1109/CTS.2014.6867557.

[Dark Reading, 12] darkreading. (2012). A Case Study In Security Big Data Analysis. Available online: https://www.darkreading.com/security-monitoring/a-case-study-in-security-big-data-analysis (last accessed December 25, 2021)

[Das Sarma, 08] Das Sarma, Anish & Dong, Xin & Halevy, Alon. (2008). Bootstrapping Pay-As-You-Go Data Integration Systems. SIGMOD. 861-874. 10.1145/1376616.1376702.

[Das Sarma, 12] Das Sarma, Anish & Fang, Lujun & Gupta, Nitin & Halevy, Alon & Lee, Hongrae & Wu, Fei & Xin, Reynold & Yu, Cong. (2012). Finding related tables. SIGMOD'12. DOI: 10.1145/2213836.2213962.

[Dasu, 03] Dasu, Tamraparni & Johnson, Theodore. (2003). Exploratory Data Mining and Data Cleaning. Canada: John Wiley & Sons.

[Dean, 04] Dean, Jeffrey & Ghemawat, Sanjay. (2004). MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM. 51. 137-150. DOI: 10.1145/1327452.1327492.

[Dean, 14] Dean, Jeffrey & Ghemawat, Sanjay. (2014). Big Data, Data Mining, and Machine Learning. John Wiley & Sons. ISBN: 9781118618042.

[Delepet, 09] Delepet, Ray & Monica, Santa. (2009). Quantifying a datasource's reputation. United States - Patent Application Publication. Pub. No. US 2009/0125382 A1. Available online: https://patentimages.storage.googleapis.com/8b/7f/e3/a2dcf7449e0eaf/US20160194368A1.pdf (last accessed December 25, 2021)

[Dincer, 17] Dincer, Cem & Zeydan, Engin. (2017). Big Data Security: Requirements, Challenges and Preservation of Private Data inside Mobile Operators. DOI: 10.1109/BlackSeaCom.2017.8277711.

[Dissanayake, 21] Dissanayake, Anusha. (2021). Big Data Security Challenges and Prevention Mechanisms in Business. International Journal of Scientific Research & Engineering Trends. Volume 7. Issue 1. issn: 2395-566X.

[Do, 02] Do, Hong-Hai & Melnik, Sergey & Rahm, Erhard. (2002). Comparison of Schema Matching Evaluations. Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems. DOI: 10.1007/3-540-36560-5_17.

[Dong, 09] Dong, Xin & Berti-Equille, Laure & Srivastava, Divesh. (2009). Integrating conflicting data. Proceedings of the VLDB Endowment. 2. 550-561. DOI: 10.14778/1687627.1687690.

[Dong, 13] Dong, Xin & Srivastava, Divesh. (2013). Big Data Integration. Proceedings of the VLDB Endowment. 6. 1245-1248. 10.1109/ICDE.2013.6544914.

[Dong, 15] Dong, Xin Luna & Srivastava, Divesh. (2015). Big Data Integration. Synthesis Lectures on Data Management. ISBN: 978-1-62705-223-8. DOI: 10.2200/S00578ED1V01Y201404DTM040.

[Dwork, 93] Dwork, Cynthia & Naor, Moni. (1993). Pricing via Processing or Combatting Junk Mail. Advances in Cryptology - CRYPT0 '92, LNCS 740, pp. 139-14, 1993. DOI: 10.1007/3-540-48071-4_10.

[El Mokhtari, 21] el Mokhtari, Jihane & Abou el Kalam, Anas & Benhadou, Siham & LEROY, Jean-Philippe. (2021). Dynamic Management of Security Policies in PrivOrBAC. International Journal of Advanced Computer Science and Applications. 12. DOI: 10.14569/IJACSA.2021.0120681.

[Erl, 16] Erl, Thomas & Khattak, Wajid & Buhler, Paul. (2016). Big Data Fundamentals - Concepts, Drivers & Techniques. Arcitura Education Inc. ISBN: 978-0-13-429107-9.

[Esteban, 05] Esteban, Jaime & Starr, Andrew & Willetts, Robert & Hannah, Paul & Bryanston-cross, Peter. (2005). A Review of data fusion models and architectures: towards engineering guidelines. Neural Computing and Applications. 14. 273-281. DOI: 10.1007/s00521-004-0463-7.

[Fellegi, 69] Fellegi, Ivan & Sunter, Alan. (1969). A Theory for Record Linkage. Journal of the American Statistical Association. 64. 1183-1210. DOI: 10.1080/01621459.1969.10501049.

[Ferraiolo, 01] Ferraiolo, David & Sandhu, Ravi & Gavrila, Serban & Kuhn, D. & Chandramouli, Ramaswamy. (2001). Proposed NIST Standard for Role Based Access Control. ACM Trans. Inf. Syst. Secur.. 4. 224-274. DOI: 10.1145/501978.501980.

[Ferraiolo, 16] Ferraiolo, David & Chandramouli, Ramaswamy & Kuhn, D. & Hu, Vincent. (2016). Extensible Access Control Markup Language (XACML) and Next Generation Access Control (NGAC). 13-24. DOI: 10.1145/2875491.2875496.

[Ferraiolo, 18] Ferraiolo, David & Gavrila, Serban & Katwala, Gopi & Roberts, Joshua. (2018). Next Generation Access Control System and Process for Controlling Database Access. United States Patent. Patent No.: US 10,127,

393 B2. Available online: https://www.nist.gov/system/files/documents/2020/02/26/15-020US1.pdf (last accessed December 25, 2021)

[FIPS PUBS, 01] Federal Information Processing Standards Publications 197. (2001). ADVANCED ENCRYPTION STANDARD (AES).

[FIPS PUBS, 99] Federal Information Processing Standards Publications 46-3. (1999). DATA ENCRYPTION STANDARD (DES).

[Firmani, 15] Firmani, Donatella & Mecella, Massimo & Scannapieco, Monica & Batini, Carlo. (2016). On the Meaningfulness of "Big Data Quality" (Invited Paper). Data Science and Engineering. 1. DOI: 10.1007/s41019-015-0004-7.

[Fox, 94] Fox, Christopher & Levitin, Anany & Redman, Thomas. (1994). The notion of data and its quality dimensions. Information Processing & Management. 30. 9-19. DOI: 10.1016/0306-4573(94)90020-5.

[Fox, 99] Fox, Armando & Brewer, Eric. (1999). Harvest, yield, and scalable tolerant systems. Proceedings of the Seventh Workshop on Hot Topics in Operating Systems. 174 - 178. DOI: 10.1109/HOTOS.1999.798396.

[Fugini, 02] Fugini, Maria & Mecella, Massimo & Plebani, Pierluigi & Scannapieco, Monica. (2002). Data Quality in Cooperative Web Information Systems. Kluwer Academic Publishers.

[Fürber, 11] Fürber, Christian & Hepp, Martin. (2011). Towards a Vocabulary for Data Quality Management in Semantic Web Architectures. Proceedings of the 1st International Workshop on Linked Web Data Management. DOI: 10.1145/1966901.1966903.

[Galland, 10] Galland, Alban & Abiteboul, Serge & Marian, Amelie & Senellart, Pierre. (2010). Corroborating Information from Disagreeing Views ∗ ABSTRACT. International Conference on Web Search and Data Mining (WSDM). DOI: 10.1145/1718487.1718504.

[Gamble, 11] Gamble, Matthew & Goble, Carole. (2011). Quality, Trust, and Utility of Scientific Data on the Web: Towards a Joint Model. Proceedings of the 3rd International Web Science Conference, WebSci 2011. DOI: 10.1145/2527031.2527048.

[Gani, 15] Gani, Abdullah & Siddiqa, Aisha & Band, Shahab & Nasaruddin, Fariza. (2015). A survey on Indexing Techniques for Big Data: Taxonomy and Performance Evaluation. Knowledge and Information Systems. 46. DOI: 10.1007/s10115-015-0830-y.

[Gantz, 11] Gantz, John & Reinsel, David. (2011). Extracting Value from Chaos. International Data Corporation (IDC).

[Gartner, 21] Gartner. (2021). Gartner Glossary - Big Data. Available online: https://www.gartner.com/en/information-technology/glossary/big-data (last accessed December 25, 2021)

[Gavrila, 98] Gavrila, Serban & Barkley, John. (1998). Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management. Proceedings of the third ACM workshop on Role-based access control. 81-90. DOI: 10.1145/286884.286902.

[Ge, 18] Ge, Mouzhi & Dohnal, Vlastislav. (2018). Quality Management in Big Data. Informatics. 5. 19. DOI: 10.3390/informatics5020019.

[Ghemawat, 03] Ghemawat, Sanjay & Gobioff, Howard & Leung, Shun-Tak. (2003). The Google File System. ACM SIGOPS Operating Systems Review. 37. 29-43. DOI: 10.1145/945445.945450.

[Glavic, 14] Glavic, Boris. (2014). Big Data Provenance: Challenges and Implications for Benchmarking. Springer, Berlin, Heidelberg. ISBN: 978-3-642-53973-2. DOI: 10.1007/978-3-642-53974-9_7.

[Global Pulse, 12] United Nations Global Pulse. (2012). Big Data for Development: Challenges & Opportunities.

[Gouglidis, 12] Gouglidis, Antonios & Mavridis, Ioannis. (2012). DomRBAC: An access control model for modern collaborative systems. Computers & Security. DOI: 10.1016/j.cose.2012.01.010.

[Goyal, 06] Goyal, Vipul & Pandey, Omkant & Sahai, Amit & Waters, Brent. (2006). Attribute-based encryption for fine-grained access control of encrypted data. Proceedings of the ACM Conference on Computer and Communications Security. 89-98. DOI: 10.1145/1180405.1180418.

[Gray, 05] Gray, Jim & Liu, David & Nieto-Santisteban, Maria & Szalay, Alexander & DeWitt, David & Heber, Gerd. (2005). Scientific Data Management in the Coming Decade. SIGMOD Record. 34. DOI: 10.1145/1107499.1107503.

[Gschwandtner, 12] Gschwandtner, Theresia & Gaertner, Johannes & Aigner, Wolfgang & Miksch, Silvia. (2012). A Taxonomy of Dirty Time-Oriented Data. International Cross-Domain Conference and Workshop on Availability, Reliability, and Security (CD-ARES). DOI: 10.1007/978-3-642-32498-7_5.

[Gupta, 17] Gupta, Maanak & Patwa, Farhan & Sandhu, Ravi. (2017). Object-Tagged RBAC Model for the Hadoop Ecosystem. 63-81. DOI: 10.1007/978-3-319-61176-1_4.

[Gupta, 18] Gupta, Maanak & Patwa, Farhan & Sandhu, Ravi. (2018). An Attribute-Based Access Control Model for Secure Big Data Processing in Hadoop Ecosystem. 13-24. DOI: 10.1145/3180457.3180463.

[Gupta, 20] Gupta, Maanak & Awaysheh, Feras & Benson, James & Alazab, Mamoun & Patwa, Farhan & Sandhu, Ravi. (2020). An Attribute-Based Access Control for Cloud Enabled Industrial Smart Vehicles. IEEE Transactions on Industrial Informatics. DOI: 10.1109/TII.2020.3022759.

[Habib, 11] Habib, Sheikh & Ries, Sebastian & Mühlhäuser, Max. (2011). Towards a Trust Management System for Cloud Computing. International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11, DOI: 10.1109/TrustCom.2011.129.

[Hakuta, 14] Hakuta, Keisuke & Sato, Hisayoshi. (2014). Cryptographic Technology for Benefiting from Big Data. Springer, The Impact of Applications on Mathematics 2014. 85-95. DOI: 10.1007/978-4-431-54907-9_6.

[Harrison, 76] Harrison, Michael & Ruzzo, Walter & Ullman, Jeffrey. (1976). On Protection in Operating System. ACM SIGOPS Operating Systems Review. 9. 14-24. DOI: 10.1145/800213.806517.

[Hashem, 14] Hashem, Ibrahim & Yaqoob, Ibrar & Anuar, Nor & Mokhtar, Salimah & Gani, Abdullah & Khan, Samee. (2014). The rise of "Big Data" on cloud computing: Review and open research issues. Information Systems. 47. 98-115. DOI: 10.1016/j.is.2014.07.006.

[Herzog, 07] Herzog, Thomas N. & Scheuren, Fritz J. & Winkler, William E.. (2007). Data Quality and Record Linkage. Springer Science+Business Media. ISBN: 978-0-387-69502-0. DOI: 10.1007/0-387-69505-2.

[HIPAA, 96] Health Insurance Portability and Accountability Act of 1996 (HIPAA). Available online: https://www.cdc.gov/phlp/publications/topic/hipaa.html (last accessed December 25, 2021)

[Hoeren, 18] Hoeren, Thomas. (2018). Big Data and Data Quality. Big Data in Context. SpringerBriefs in Law. DOI: 10.1007/978-3-319-62461-7_1.

[Hu, 15] Hu, Vincent & Kuhn, D. & Ferraiolo, David. (2015). Attribute-Based Access Control. Computer. Volume 48. Issue 2. Pages 85-88. DOI: 10.1109/MC.2015.33.

[Hussain, 07] Hussain, Farookh & Hussain, Omar & Chang, Elizabeth. (2007). An overview of the interpretations of trust and reputation. IEEE International Conference on Emerging Technologies and Factory Automation, ETFA. 826-830. DOI: 10.1109/EFTA.2007.4416865.

[ICO, 14] Information Commissioner's Office (ICO). (2014). Big data and data protection. Data Protection Act 1998.

[Immonen, 15] Immonen, Anne & Pääkkönen, Pekka & Ovaska, Eila. (2015). Evaluating the Quality of Social Media Data in Big Data Architecture. IEEE Access. 3. 1-1. DOI: 10.1109/ACCESS.2015.2490723.

[InfoWorld, 13] InfoWorld. (2013). THINK BIG DATA - The real story of how big data analytics helped Obama win. Available online: https://www.infoworld.com/article/2613587/the-real-story-of-how-big-data-analytics-helped-obama-win.html (last accessed December 25, 2021)

[Inukollu, 14] Inukollu, Venkata & Arsi, Sailaja & Ravuri, Srinivasa. (2014). Security Issues Associated with Big Data in Cloud Computing. International Journal of Network Security & Its Applications. 6. 45-56. DOI: 10.5121/ijnsa.2014.6304.

[ISO-IEC 25010, 08] ISO. ISO/IEC 25010. (2008). CD 25010.2, Software engineering-Software product Quality Requirements and Evaluation (SQuaRE)Quality model.

[ISO-IEC 25012, 06] ISO. ISO/IEC 25012. (2006). CD 25012 Software engineering: Software Quality Requirements and Evaluation (SQuaRE) - Data Quality Model.

[ISO-IEC 25024, 15] ISO. ISO/IEC 25024. (2015). Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of data quality.

[ISO-IEC 27001, 05] ISO. ISO/IEC 27001. (2005). Information technology — Security techniques — Information security management systems — Requirements.

[Jakobsson, 99] Jakobsson, Markus & Juels, Ari. (1999). Proofs of Work and Bread Pudding Protocols.. Communications and Multimedia Security. 258-272. DOI: 10.1007/978-0-387-35568-9_18.

[Jaro, 89] Jaro, Matthew. (1989). Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. Journal of The American Statistical Association - J AMER STATIST ASSN. 84. 414-420. DOI: 10.1080/01621459.1989.10478785.

[Jensen, 13] Jensen, Meiko. (2013). Challenges of Privacy Protection in Big Data Analytics. IEEE International Congress on Big Data. 235-238. DOI: 10.1109/BigData.Congress.2013.39.

[Kaisler, 13] Kaisler, Stephen & Armour, Frank & Espinosa, J. & Money, William. (2013). Big Data: Issues and Challenges Moving Forward. Proceedings of the Annual Hawaii International Conference on System Sciences. 995-1004. DOI: 10.1109/HICSS.2013.645.

[Kannan, 11] Kannan, Anitha & Givoni, Inmar & Agrawal, Rakesh & Fuxman, Ariel. (2011). Matching unstructured product offers to structured product specifications. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 404-412. DOI: 10.1145/2020408.2020474.

[Kapil, 18] Kapil, Gayatri & Agrawal, Alka & Khan, Prof. Raees. (2018). Big Data Security challenges: Hadoop Perspective. International Journal of Pure and Applied Mathematics. 120. 11767-11784.

[Katal, 13] Katal, Avita & Wazid, Mohammad & Goudar, R.H.. (2013). Big data: Issues, challenges, tools and Good practices. IEEE. 404-409. DOI: 10.1109/IC3.2013.6612229.

[Khan, 14] Khan, Nawsher & Yaqoob, Ibrar & Hashem, Ibrahim & Inayat, Zakira & Kamaleldin, Waleed & Alam, Muhammad & Shiraz, Muhammad & Gani, Abdullah. (2014). Big Data: Survey, Technologies, Opportunities, and Challenges. The Scientific World Journal. 2014. 18. DOI: 10.1155/2014/712826.

[Khan, 18] Khan, Nawsher & Alsaqer, Mohammed & Shah, Habib & Badsha, Gran & Abbasi, Aftab & Salehian, Solmaz. (2018). The 10 Vs, Issues and Challenges of Big Data. 52-56. DOI: 10.1145/3206157.3206166.

[Kleiner, 14] Kleiner, Ariel & Talwalkar, Ameet & Sarkar, Purnamrita & Jordan, Michael. (2014). A Scalable Bootstrap for Massive Data. Journal of the Royal Statistical Society Series B (Statistical Methodology). 76. DOI: 10.1111/rssb.12050.

[Krithika, 20] Krithika, D.R. & Rohini. K..(2020). A Survey on Challenging Capabilities of Big Data Analytics in Healthcare. International Journal of Innovative Research in Applied Sciences and Engineering. 4. 593-597. DOI: 10.29027/IJIRASE.v4.i1.2020.593-597.

[Kuhn, 10] Kuhn, D. Richard & Coyne, Edward J. & Weil, Timothy R.. (2010). Adding Attributes to Role-Based Access Control. Computer. 43. 79-81. DOI: 10.1109/MC.2010.155.

[Labidi, 17] Labidi, Taher & Mtibaa, Achraf & Gaaloul, Walid & Gargouri, Faiez. (2017). Ontology-Based SLA Negotiation and Re-Negotiation for Cloud Computing. IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises. 36-41. DOI: 10.1109/WETICE.2017.24

[Lampson, 74] Lampson, Butler. (1974). Protection. ACM SIGOPS Operating Systems Review. 8. 18-24. DOI: 10.1145/775265.775268.

[Landset, 15] Landset, Sara & Khoshgoftaar, Taghi & Richter, Aaron & Hasanin, Tawfiq. (2015). A survey of open source tools for machine learning with big data in the Hadoop ecosystem. Journal of Big Data. 2. DOI: 10.1186/s40537-015-0032-1.

[Laranjeiro, 15] Laranjeiro, Nuno & Soydemir, Seyma & Bernardino, Jorge. (2015). A Survey on Data Quality: Classifying Poor Data. IEEE 21st Pacific Rim International Symposium on Dependable Computing 2015. 179-188. DOI: 10.1109/PRDC.2015.41.

[Lee, 17] Lee, In. (2017). Big data: Dimensions, evolution, impacts, and challenges. Business Horizons. 60. DOI: 10.1016/j.bushor.2017.01.004.

[Levenshtein, 66] Levenshtein, V.I.. (1966). Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. Soviet Physics Doklady. 10. 707-710.

[Lewko, 10] Lewko, Allison & Okamoto, Tatsuaki & Sahai, Amit & Takashima, Katsuyuki & Waters, Brent. (2010). Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption.

Advances in cryptology EUROCRYPT 2010, Lecture Notes in Computer Science. 6110. 62-91. DOI: 10.1007/978-3-642-13190-5_4.

[Lewko, 11] Lewko, Allison & Waters, Brent. (2011). Decentralizing Attribute-Based Encryption. Advances in Cryptology, EUROCRYPT 2011. 568-588. DOI: 10.1007/978-3-642-20465-4_31

[Li, 14-a] Li, Jingwei & Chen, Xiaofeng & Li, Mingqiang & Lee, Patrick & Lou, Wenjing. (2014). Secure Deduplication with Efficient and Reliable Convergent Key Management. Parallel and Distributed Systems, IEEE Transactions on. 25. 1615-1625. DOI: 10.1109/TPDS.2013.284.

[Li, 14-b] Li, Yanhuang & Cuppens-Boulahia, Nora & Crom, Jean-Michel & Cuppens, Frédéric & Frey, Vincent. (2014). Reaching Agreement in Security Policy Negotiation. IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications. DOI: 10.1109/TrustCom.2014.17.

[Li, 16] Li, Yanhuang & Cuppens-Boulahia, Nora & Crom, Jean-Michel & Cuppens, Frédéric & Frey, Vincent. (2016). Expression and Enforcement of Security Policy for Virtual Resource Allocation in IaaS Cloud. 105-118. DOI: 10.1007/978-3-319-33630-5_8.

[Li, 19-a] Li, Fan & Cabrera, Christian & Clarke, Siobhán. (2019). A WS-Agreement Based SLA Ontology for IoT Services. 4th International Conference Held as Part of the Services Conference Federation. Proceedings. DOI: 10.1007/978-3-030-23357-0_5.

[Li, 19-b] Li, Fan & Clarke, Siobhán. (2019). A Context-Based Strategy for SLA Negotiation in the IoT Environment. PerIoT'19 - Third International Workshop on Mobile and Pervasive Internet of Things. DOI: 10.1109/PERCOMW.2019.8730752.

[Libération, 17] Libération. (2017). Succès fous (4/4) - Cambridge Analytica, big data et gros dégâts. Available online: https://www.liberation.fr/futurs/2017/08/17/cambridge-analytica-big-data-et-gros-degats_1590468/ (last accessed December 25, 2021)

[Liu, 11] Liu, Xuan & Dong, Xin & Ooi, Beng & Srivastava, Divesh. (2011). Online Data Fusion.. PVLDB. 4. 932-943. DOI: 10.14778/3402707.3402731.

[Liu, 13] Liu, Xuejiao & Xia, Yingjie & Jiang, Shasha & Xia, Fubiao & Wang, Yanbo. (2013). Hierarchical Attribute-Based Access Control with Authentication for Outsourced Data in Cloud Computing. 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. 477-484. DOI: 10.1109/TrustCom.2013.60.

[Loshin, 01] Loshin, David. (2001). Enterprise Knowledge Management: The Data Quality Approach. Morgan Kaufmann Publishers Inc. ISBN: 0-12-455840-2.

[Loshin, 13] Loshin, David. (2013). Big Data Analytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL, and Graph. Morgan Kaufmann Publishers. ISBN: 9780124186644.

[LPRPDE, 18] Loi sur la protection des renseignements personnels et les documents électroniques (LPRPDE). Available online: https://www.priv.gc.ca/fr/sujets-lies-a-la-protection-de-la-vie-privee/lois-sur-la-protection-des-renseignements-personnels-au-canada/la-loi-sur-la-protection-des-renseignements-personnels-et-les-documents-electroniques-lprpde/r_o_p/ (last accessed December 25, 2021)

[Lu, 14] Lu, Rongxing & Zhu, Hui & Liu, Ximeng & Liu, Joseph & Shao, Jun. (2014). Toward Efficient and Privacy-Preserving Computing in Big Data Era. Network, IEEE. 28. 46-50. DOI: 10.1109/MNET.2014.6863131.

[Ludwig, 03] Ludwig, Heiko & Keller, Alexander & Dan, Asit & King, Richard & Franck, Richard. (2003). Web Service Level Agreement (WSLA) Language Specification. IBM Corporation. 815–824.

[Ludwig, 06] Ludwig, Heiko & Nakata, Toshiyuki & Wäldrich, Oliver & Wieder, Philipp & Ziegler, Wolfgang. (2006). Reliable Orchestration of Resources Using WS-Agreement. Gerndt M., Kranzlmüller D. (eds) High Performance Computing and Communications. HPCC 2006. Lecture Notes in Computer Science. Volume 4208. DOI: 10.1007/11847366_78.

[Lv, 16] Lv, Bin & Zhang, Di & Mao, Rui & Yang, Haitian. (2016). A Multi-Level Cross-Domain Access Control Model Based On Role Mapping. 4th International Conference on Mechanical Materials and Manufacturing Engineering (MMME 2016). DOI: 10.2991/mmme-16.2016.53.

[Maletic, 00] Maletic, Jonathan & Marcus, Andrian. (2000). Data Cleansing: Beyond Integrity Analysis. Proceedings of the Conference on Information Quality. 200-209.

[Marino, 18] Marino, Francesco & Moiso, Corrado & Petracca, Matteo. (2018). Automatic contract negotiation, service discovery and mutual authentication solutions: A survey on the enabling technologies of the forthcoming IoT ecosystems. Computer Networks. 148. DOI: 10.1016/j.comnet.2018.11.011.

[Marz, 15] Marz, Nathan & Warren, James. (2015). Big Data - Principles and best practices of scalable real-time data systems. Manning Publications Co. ISBN: 9781617290343.

[Matturdi, 14] Matturdi, Bardi & Zhou, Xianwei & Li, Shuai & Lin, Fuhong. (2014). Big Data security and privacy: A review. China Communications. 11. 135-145. DOI: 10.1109/CC.2014.7085614.

[Mayol, 99] Mayol, Enric & Teniente, Ernest. (1999). A Survey of Current Methods for Integrity Constraint Maintenance and View Updating. DOI: 10.1007/3-540-48054-4_6.

[Merino, 15] Merino, Jorge & Caballero, Ismael & Rivas Garcia, Bibiano & Serrano, Manuel & Piattini, Mario. (2015). A Data Quality in Use model for Big Data. Future Generation Computer Systems. 63. DOI: 10.1016/j.future.2015.11.024.

[meta, 18] meta. (2018). An Update on Our Plans to Restrict Data Access on Facebook. Available online: https://about.fb.com/news/2018/04/restricting-data-access/ (last accessed December 25, 2021)

[Missier, 03] Missier, Paolo & Lalk, G. & Verykios, Vassilios & Grillo, F. & Lorusso, T. & Angeletti, P.. (2003). Improving Data Quality in Practice: A Case Study in the Italian Public Administration. Distributed and Parallel Databases. 13. 135-160. DOI: 10.1023/A:1021548024224.

[Moore, 08] Moore, Adam. Defining Privacy. (2008). Journal of social philosophy. Volume 39. No. 3. 411–428.

[Motro, 98] Motro, Amihai & Rakov, Igor. (1998). Estimating the Quality of Databases. Springer-Verlag Berlin Heidelberg 1998.

[Müller, 03] Müller, Heiko & Freytag, Johann-Christoph. (2003). Problems, methods, and challenges in comprehensive data cleansing. Technical Report HUB-IB-164. Humboldt University.

[Mylavarapu, 19] Mylavarapu, Goutam & Thomas, Johnson & Kannan, Ashwin. (2019). An Automated Big Data Accuracy Assessment Tool. IEEE 4th International Conference on Big Data Analytics (ICBDA). 193-197. DOI: 10.1109/ICBDA.2019.8713218.

[Nakamoto, 08] Nakamoto, Satoshi. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: https://bitcoin.org/bitcoin.pdf (last accessed December 25, 2021)

[Naumann, 99] Naumann, Felix & Leser, Ulf & Freytag, Johann Christoph. (1999). Quality-driven Integration of Heterogeneous Information Systems. Proceedings of the 25th VLDB Conference.

[Newcombe, 59] Newcombe, H. & Kennedy, J. & AXFORD, S & James, AP. (1959). Automatic Linkage for Vital Records. Science. 130. 954-959. DOI: 10.1126/science.130.3381.954.

[Olensky, 14] Olensky, Marlies. (2014). Testing an Automated Accuracy Assessment Method on Bibliographic Data. Journal of Library and Information Studies. 12. DOI: 10.6182/jlis.2014.12(2).019.

[Oliveira, 05] Oliveira, Paulo & Rodrigues, Fátima & Rangel Henriques, Pedro. (2005). A Formal Definition of Data Quality Problems.. Proceedings of the 2005 International Conference on Information Quality, ICIQ 2005.

[ONU, 48] Organisation des Nations Unies - ONU. (1948). La Déclaration universelle des droits de l'homme. Available online: https://www.un.org/fr/universal-declaration-human-rights/ (last accessed December 25, 2021)

[Ouaddah, 17] Ouaddah, Aafaf & Elkalam, Anas & Ouahman, Abdellah. (2017). Towards a Novel Privacy-Preserving Access Control Model Based on Blockchain Technology in IoT. DOI: 10.1007/978-3-319-46568-5_53.

[Oussous, 17] Oussous, Ahmed & Benjelloun, Fatima-Zahra & Ait Lahcen, Ayoub & Belfkih, Samir. (2017). Big Data Technologies: A Survey. Journal of King Saud University - Computer and Information Sciences. DOI: 10.1016/j.jksuci.2017.06.001.

[Park, 13] Park, Seonyoung & Lee, Youngseok. (2013). Secure Hadoop with Encrypted HDFS. International Conference on Grid and Pervasive Computing. Grid and Pervasive Computing. pp 134-141. DOI: 10.1007/978-3-642-38027-3_14

[Parmar, 17] Parmar, Raj & Roy, Sudipta & Bhattacharaya, Debnath & Bandyopadhyay, Samir & Kim, Tai-hoon. (2017). Large Scale Encryption in Hadoop Environment: Challenges and Solutions. IEEE Access. PP. 1-1. DOI: 10.1109/ACCESS.2017.2700228.

[Pasternack, 10] Pasternack, Jeff & Roth, Dan. (2010). Knowing What to Believe (when you already know something). Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010).

[Pathak, 17] Pathak, Pankaj & Asstt, Sr & Professor, & Vyas, Nitesh & Professor, Asstt & Joshi, Someshwar. (2017). Security Challenges for Communications on IOT & Big Data. International Journal of Advanced Research in Computer Science. 8.

[Peralta, 06] Peralta, Veronika. (2006). Data Quality Evaluation in Data Integration Systems. Thesis. Université de Versailles Saint-Quentin-en-Yvelines (France) and Universidad de la República (Uruguay)

[Pinno, 17] Pinno, Otto Julio & Grégio, André & Bona, Luis Carlos. (2017). ControlChain: Blockchain as a Central Enabler for Access Control Authorizations in the IoT. University of Paris 2 Panth´eon-Assas, CRED Paris Center for Law and Economics, Chaire Finance Digitale.. DOI: 10.1109/GLOCOM.2017.8254521.

[Pipino, 02] Pipino, Leo L. & Lee, Yang W. & Wang, Richard Y.. (2002). Data Quality Assessment. Communications of the ACM. 45. DOI: 10.1145/505248.506010.

[Prajapati, 13] Prajapati, Vignesh. (2013). Big Data Analytics with R and Hadoop. Birmingham: Packt Publishing. ISBN: 978-1782163282.

[Pyne, 16] Pyne, Saumyadipta & Rao, B.L.S. & Rao, S.B.. (2016). Big data analytics: Methods and applications. Springer India 2016. ISBN: 978-81-322-3626-9. DOI: 10.1007/978-81-322-3628-3.

[Rafique, 12] Rafique, Irfan & Lew, P. & Abbasi, M.Q. & Li, Z.. (2012). Information quality evaluation framework: Extending ISO 25012 data quality model. World academy of science. Engineering and Technology. 65. 523-528.

[Rahm, 00] Rahm, Erhard & Do, Hong. (2000). Data Cleaning: Problems and Current Approaches. IEEE Data Eng. Bull.. 23. 3-13.

[Rahm, 01] Rahm, Erhard & Bernstein, Philip. (2001). A Survey of Approaches to Automatic Schema Matching.. VLDB J.. 10. 334-350. DOI: 10.1007/s007780100057.

[Rahm, 11] Rahm, Erhard. (2011). Towards Large-Scale Schema and Ontology Matching. Springer-Verlag Berlin Heidelberg 2011. DOI: 10.1007/978-3-642-16518-4_1.

[Raj, 18] Raj, Pethuru. (2018). The Hadoop Ecosystem Technologies and Tools. Advances in Computers, Volume 109. DOI: 10.1016/bs.adcom.2017.09.002.

[Raman, 00] Raman, Vijayshankar & Hellerstein, Joseph M.. (2000). An Interactive Framework for Data Cleaning. Report No. UCB/CSD-0-1110. Computer Science Division (EECS). University of California. Berkeley, California 94720.

[Ray, 04] Ray, Indrakshi & Li, Na & France, Robert & Kim, Dae-Kyoo. (2004). Using UML to visualize role-based access control constraints. Proceedings of ACM Symposium on Access Control Models and Technologies (SACMAT 2002). 9. 115-124. DOI: 10.1145/990036.990054.

[Redman, 05] Redman, Thomas C.. (2005). Measuring Data Accuracy: A Framework and Review. Information Quality. Taylor & francis group. 21-36. ISBN: 9781317467991.

[Redman, 13] Redman, Thomas C.. (2013). Data's Credibility Problem. Harvard Business Review. R1312E.

[Redman, 96] Redman, Thomas C.. (1996). Data Quality for the Information Age. Artech House. ISBN: 9780890068830.

[Reinsel, 17] Reinsel, David & Gantz, John & Rydning, John. (2017). Data Age 2025: The Evolution of Data to Life-Critical; Don't Focus on Big Data; Focus on the Data That's Big. IDC White Paper.

[Reuters, 17] Reuters. (2017). TECHNOLOGY NEWS. EU seeks to expedite police requests for data from tech firms. Available online: https://www.reuters.com/article/us-eu-data-security-idUSKBN18Z0H0 (last accessed December 25, 2021)

[RezaeiJam, 14] RezaeiJam, Masoumeh & Khanli, Leili Mohammad & Akbari, Mohammad Kazem & Javan, Morteza Sargolzaei. (2014). A survey on security of Hadoop. Proceedings of the 4th International Conference on Computer and Knowledge Engineering, ICCKE 2014. 716-721. DOI: 10.1109/ICCKE.2014.6993455.

[Rissanen, 17] Rissanen, Erik. (2017). eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01. OASIS Standard incorporating Approved Errata. Available online: http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os.pdf (last accessed December 25, 2021)

[Rivest, 92] Rivest, R.. (1992). The MD5 Message-digest Algorithm. Network Working Group. MIT Laboratory for Computer Science and RSA Data Security.

[RSA, 78] Rivest, Ron & Shamir, Adi & Adleman, Len. (1978). A Method for Obtaining Digital Signatures and PublicKey Cryptosystems. Communications of the ACM. Volume 21. Issue 2. pp 120–126. DOI: 10.1145/359340.359342.

[Saha, 14] Saha, Barna & Srivastava, Divesh. (2014). Data quality: The other face of Big Data. Proceedings - International Conference on Data Engineering. 1294-1297. DOI: 10.1109/ICDE.2014.6816764.

[Sahai, 05] Sahai, Amit & Waters, Brent. (2005). Fuzzy Identity Based Encryption. Lecture Notes in Computer Science. 3494. DOI: 10.1007/11426639_27.

[Saleh, 14] Saleh, Aya Salama A. & Hamed, Essam M. Ramzy & Hashem, Mohamed. (2014). Building trust management model for cloud computing. 2014 9th International Conference on Informatics and Systems, INFOS 2014. PDC116-PDC125. DOI: 10.1109/INFOS.2014.7036688.

[Sandhu, 93] Sandhu, Ravi S.. (1993). Lattice-based access control models. IEEE. Computer. Volume 26. pp. 9-19. DOI: 10.1109/2.241422.

[Sandhu, 96] Sandhu, Ravi S. & Coynek, Edward J. & Feinsteink, Hal L. & Youmank, Charles E.. (1996). Role-Based Access Control Models. IEEE Computer. Volume 29. Number 2. pages 38-47.

[Sangeetha, 15] Sangeetha, S. & Sreeja, A.K. (2015). No Science No Humans, No new Technologies No changes "Big Data A Great Revolution". International Journal of Computer Science and Information Technologies, Volume 6. Issue 4. 3269-3274.

[Scannapieco, 05] Scannapieco, Monica & Missier, Paolo & Batini, Carlo. (2005). Data Quality at a Glance.. Datenbank-Spektrum. 14. 6-14.

[Schneier, 05] Schneier. (2005). Schneier on Security. Cryptanalysis of SHA-1. Available online: https://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html (last accessed December 25, 2021)

[Scoca, 17] Scoca, Vincenzo & Uriarte, Rafael Brundo & De Nicola, Rocco. (2017). Smart Contract Negotiation in Cloud Computing. IEEE 10th International Conference on Cloud Computing. DOI: 10.1109/CLOUD.2017.81.

[Seng, 19] Seng, Kah & Ang, Li-Minn. (2019). Multimodal Emotion and Sentiment Modeling From Unstructured Big Data: Challenges, Architecture, & Techniques. IEEE Access. PP. 1-1. DOI: 10.1109/ACCESS.2019.2926751.

[Shankaranarayanan, 03] Shankaranarayanan, Ganesan & Ziad, Mostapha & Wang, Richard. (2003). Managing Data Quality in Dynamic Decision Environments. Journal of Database Management. 14. 14-32. DOI: 10.4018/jdm.2003100102.

[Shanks, 99] Shanks, Graeme & Corbitt, Brian. (1999). Understanding data quality: Social and cultural aspects. Proceedings of the 10th Australasian Conference on Information Systems.

[Sherman, 92] Sherman, Robin L.. (1992). Distributed systems security. Computers and Security. Volume 11. Issue 1. pp 24–28. DOI: 10.1016/0167-4048(92)90216-E.

[Shojaiemehra, 19] Shojaiemehr, Bahador & Rahmani, Amir & Qader, Nooruldeen. (2019). A Three-phase Process for SLA Negotiation of Composite Cloud Services. Computer Standards & Interfaces. 64. DOI: 10.1016/j.csi.2019.01.001.

[Shvaiko, 05] Shvaiko, Pavel & Euzenat, Jerome. (2005). A Survey of Schema-Based Matching Approaches. Lecture Notes in Computer Science. DOI: 10.1007/11603412_5.

[Shvaiko, 13] Shvaiko, Pavel & Euzenat, Jérôme. (2013). Ontology matching: state of the art and future challenges. IEEE Transactions on Knowledge and Data Engineering, Institute of Electrical and Electronics Engineers. pp.158-176. DOI: 10.1109/TKDE.2011.253ff.

[Singh, 18] Singh, Vikash Kumar & Taram, Manish & Agrawal, Vinni & Baghel, Bhartee Singh. (2018). A Literature Review on Hadoop Ecosystem and Various Techniques of Big Data Optimization. Springer Nature Singapore Pte Ltd. Advances in Data and Information Sciences, Lecture Notes in Networks and Systems 38. DOI: 10.1007/978-981-10-8360-0_22.

[SLA-Framework, 16] SLA-Framework. (2016). Version 1.1. Available online: https://github.com/FIWARE/ops.Sla-framework (last accessed December 25, 2021)

[Smith, 07] Smith, Matthew & Schmidt, M & Fallenbeck, Niels & Schridde, C & Freisleben, B. (2007). Optimising Security Configurations with Service Level Agreements. Department of Mathematics and Computer Science, University of Marburg, Germany.

[Stankov, 12] Stankov, I. & Datsenka, Rastsislau & Kurbel, K.. (2012). Service level agreement as an instrument to enhance trust in cloud computing - An analysis of infrastructureas-a-service providers. 18th Americas Conference on Information Systems 2012, AMCIS 2012. 5. 3813-3822.

[Strong, 97] Strong, Diane & Lee, Yang & Wang, Richard. (2002). Data Quality in Context. Communications of the ACM. 40. DOI: 10.1145/253769.253804.

[Strozzi, 98] Strozzi. (1998). NoSQL: a non-SQL RDBMS. Available online: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page (last accessed December 25, 2021)

[Stvilia, 07] Stvilia, Besiki & Gasser, Les & Twidale, Michael & Smith, Linda. (2007). A framework for information quality assessment. JASIST. 58. 1720-1733. DOI: 10.1002/asi.20652.

[Suarez, 92] Suarez, J. Gerald. (1992). Three Experts on Quality Management: Philip B. Crosby, W. Edwards Deming, Joseph M. Juran. TQLO Publication No. 92-02.

[Sudarsan, 15] Sudarsan, Sithu & Jetley, Raoul & Ramaswamy, Srini. (2015). Security and Privacy of Big Data. In book: Big Data (ISBN: 978-81-322-2493-8. pp.121-136). DOI: 10.1007/978-81-322-2494-5_5.

[Sun, 11] Sun, Xiaodong & Chang, Guiran & Li, Fengyun. (2011). A Trust Management Model to Enhance Security of Cloud Computing Environments. Proceedings - 2nd International Conference on Networking and Distributed Computing, ICNDC 2011. 244 - 248. DOI: 10.1109/ICNDC.2011.56.

[Sutanta, 16] Sutanta, Edhy. (2016). Survey: Models and Prototypes of Schema Matching. International Journal of Electrical & Computer Engineering. 6. DOI: 10.11591/ijece.v6i3.9789.

[Taleb, 16] Taleb, Ikbal & El Kassabi, Hadeel & Serhani, Mohamed & Dssouli, Rachida & Bouhaddioui, Chafik. (2016). Big Data Quality: A Quality Dimensions Evaluation. Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress. DOI: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.145.

[Taleb, 18] Taleb, Ikbal & Serhani, Mohamed & Dssouli, Rachida. (2018). Big Data Quality: A Survey. IEEE International Congress on Big Data. DOI: 10.1109/BigDataCongress.2018.00029.

[Talha, 19] Talha, Mohamed & Abou El Kalam, Anas & Elmarzouqi, Nabil. (2019). Big Data: Trade-off between Data Quality and Data Security. The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019). April 29 - May 2, 2019, Leuven, Belgium. Procedia Computer Science. 151. 916-922. DOI: 10.1016/j.procs.2019.04.127.

[Talha, 20-a] Talha, Mohamed & Elmarzouqi Nabil & Abou El Kalam, Anas. (2020). Quality and Security in Big Data: Challenges as opportunities to build a powerful wrap-up solution. Journal of Ubiquitous Systems and Pervasive Networks. 12. 09-15. DOI: 10.5383/JUSPN.12.01.002.

[Talha, 20-b] Talha, Mohamed & Elmarzouqi, Nabil & Abou El Kalam, Anas. (2020). Towards a Powerful Solution for Data Accuracy Assessment in the Big Data Context. International Journal of Advanced Computer Science and Applications. 11. DOI: 10.14569/IJACSA.2020.0110254.

[Talha, 20-c] Talha, Mohamed & El Mokhtari, Jihane & Abou El Kalam, Anas. (2020). PACT: Privacy Aware Contact-Tracing - Big Data Security and Privacy in the COVID-19 Era. Journal of Multidisciplinary Engineering Science and Technology (JMEST). Volume 7. Issue 5. ISSN: 2458-9403.

[Talha, 21-a] Talha, Mohamed & Abou El Kalam, Anas. (2021). Big Data between Quality and Security: Dynamic Access Control for Collaborative Platforms. JUCS - Journal of Universal Computer Science 27(12): 1300-1324. DOI: 10.3897/jucs.77046.

[Talha, 21-b] Talha, Mohamed & Abou El Kalam, Anas. (2022). Big Data: towards a collaborative security system at the service of data quality. The 17th International Conference on Information Assurance and Security (IAS 2021). 14-16 December 2021 on the World Wide Web. HIS 2022, LNNS 420, pp. 1–12, 2022. Hybrid Intelligent Systems. Chapter 55. ISBN: 978-3-030-96304-0. DOI: 10.1007/978-3-030-96305-7_55

[The White House, 12] The White House. (2012). PRESS RELEASE: Obama Administration Unveils "Big Data" Initiative: Announces $200 Million in New R&D Investments. Available online: https://obamawhitehouse.archives.gov/the-press-office/2015/11/19/release-obama-administration-unveils-big-data-initiative-announces-200 (last accessed December 25, 2021)

[Tolone, 05] Tolone, William & Ahn, Gail-Joon & Pai, Tanusree & Hong, Seng-Phil. (2005). Access control in collaborative systems. ACM Comput. Surv.. 37. 29-41. DOI: 10.1145/1057977.1057979.

[Toshniwal, 15] Toshniwal, Raghav & Dastidar, Kanishka & Nath, Asoke. (2015). Big Data Security Issues and Challenges. International Journal of Innovative Research in Advanced Engineering. 2. 15-20.

[Tosic, 02] Tosic, Vladimir & Patel, Kruti & Pagurek, Bernard. (2002). WSOL — Web Service Offerings Language. Information Systems. Volume 30. 564–586, DOI: 10.1007/3-540-36189-8_5.

[Toumi, 12] Toumi, Khalifa & Cavalli, Ana & andres, Cesar. (2012). Trust-orBAC: A Trust Access Control Model in Multi-Organization Environments. Information Systems Security. ICISS 2012. Lecture Notes in Computer Science. Volume 7671. DOI: 10.1007/978-3-642-35130-3_7.

[Tsai, 16] Tsai, Chun-Wei & Lai, Chin-Feng & Chao, Han-Chieh & Vasilakos, Athanasios. V.. (2016). Big data analytics: a survey. Journal of Big Data. DOI: 10.1007/978-3-319-44550-2_2.

[Tseng, 16] Tseng, Shian-Shyong & Chen, Hsing-Chung & Hu, Li-Ling & Lin, Yen-Tsung. (2016). CBR-based negotiation RBAC model for enhancing ubiquitous resources management. International Journal of Information Management. 37. DOI: 10.1016/j.ijinfomgt.2016.05.009.

[Viscusi, 14] Viscusi, Gianluigi & Spahiu, Blerina & Maurino, Andrea & Batini, Carlo. (2014). Compliance with open government data policies: An empirical assessment of Italian local public administrations. Information Polity. 19. DOI: 10.3233/IP-140338.

[Vitter, 85] Vitter, Jeffrey Scott. (1985). Random Sampling with a Reservoir. ACM Transactions on Mathematical Software. Volume 11. 37-57. DOI: 10.1145/3147.3165.

[Wäldrich, 11] Wäldrich, Oliver & Battré, Dominic & Brazier, Frances & Clark, Kas & Oey, Michel & Papaspyrou, Alexander & Wieder, Philipp & Ziegler, Wolfgang. (2011). WS-Agreement Negotiation Version 1.0. Available online: http://www.ogf.org/documents/GFD.193.pdf (last accessed December 25, 2021)

[Wand, 96] Wand, Yair & Wang, Richard. (1996). Anchoring Data Quality Dimensions in Ontological Foundations. Commun. ACM. 39. 86-95. DOI: 10.1145/240455.240479.

[Wang, 04] Wang, Xiaoyun & Feng, Dengguo & Lai, Xuejia & Yu, Hongbo. (2004). Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.. IACR Cryptology ePrint Archive. 2004. 199.

[Wang, 13] Wang, Lizhe & Tao, Jie & Ranjan, R. & Marten, Holger & Streit, Achim & Chen, Jingying & Chen, Dan. (2013). G-Hadoop: MapReduce across distributed data centers for data-intensive computing. Future Generation Computer Systems. 29. 739–750. DOI: 10.1016/j.future.2012.09.001.

[Wang, 16] Wang, Hai & Xu, Zeshui & Fujita, Hamido & Liu, Shousheng. (2016). Towards Felicitous Decision Making: An Overview on Challenges and Trends of Big Data. Information Sciences. 367. DOI: 10.1016/j.ins.2016.07.007.

[Wang, 96] Wang, Richard & Strong, Diane. (1996). Beyond Accuracy: What Data Quality Means to Data Consumers. J. of Management Information Systems. 12. 5-33. DOI: 10.1080/07421222.1996.11518099.

[Waters, 11] Waters, Brent. (2011). Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. International Association for Cryptologic Research 2011. PKC 2011, LNCS 6571, pp. 53–70.

[Wei, 16] Wei, Jianghong & Liu, Wenfen & Hu, Xuexian. (2016). Secure and Efficient Attribute-Based Access Control for Multiauthority Cloud Storage. IEEE Systems Journal. PP. 1-12. DOI: 10.1109/JSYST.2016.2633559.

[Winkler, 90] Winkler, William. (1990). String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. Proceedings of the Section on Survey Research Methods.

[wsag4j, 12] WSAG4J framework. (2012). Version 2.0.0. Available online: http://wsag4j.sourceforge.net/site/wsag/overview.html (last accessed December 25, 2021)

[Xiao, 15] Xiao, Min & Wang, Mingxin & Liu, Xuejiao & Sun, Junmei. (2015). Efficient distributed access control for big data in clouds. The Third International Workshop on Security and Privacy in Big Data (BigSecurity 2015). 202-207. DOI: 10.1109/INFCOMW.2015.7179385.

[Xu, 12] Xu, Lei & Wu, Xiaoxin & Zhang, Xinwen. (2012). CL-PRE: A Certificateless Proxy Re-encryption Scheme for Secure Data Sharing with Public Cloud. Proc. 7th ACM Symp. on Information, Computer and Communications Security. 87-88. DOI: 10.1145/2414456.2414507.

[Yang, 14] Yang, Kan & Jia, Xiaohua & Ren, Kaili & Xie, Ruitao & Huang, Liusheng. (2014). Enabling efficient access control with dynamic policy updating for big data in the cloud. Proceedings - IEEE INFOCOM. 2013-2021. DOI: 10.1109/INFOCOM.2014.6848142.

[Yang, 20] Yang, Min. (2020). TDACS: an ABAC and Trust-based Dynamic Access Control Scheme in Hadoop. arXiv:2011.07895.

[Yin, 08] Yin, Xiaoxin & Han, Jiawei & Yu, Philip. (2008). Truth Discovery with Multiple Conflicting Information Providers on the Web. Knowledge and Data Engineering, IEEE Transactions on. 20. 796 - 808. DOI: 10.1109/TKDE.2007.190745.

[Yuan, 05] Yuan, Eric & Tong, J.. (2005). Attributed Based Access Control (ABAC) for web services. Proceedings - 2005 IEEE International Conference on Web Services, ICWS 2005. 2005. 569. DOI: 10.1109/ICWS.2005.25.

[Yujian, 07] Yujian, Li & Bo, Liu. (2007). A Normalized Levenshtein Distance Metric. IEEE transactions on pattern analysis and machine intelligence. 29. 1091-5. DOI: 10.1109/TPAMI.2007.1078.

[Zhang, 18-a] Zhang, Lili & Xie, Yuxiang & Xidao, Luan & Zhang, Xin. (2018). Multi-source heterogeneous data fusion. International Conference on Artificial Intelligence and Big Data (ICAIBD). 47-51. DOI: 10.1109/ICAIBD.2018.8396165.

[Zhang, 18-b] Zhang, Yinghui & Zheng, Dong & Deng, Robert. (2018). Security and Privacy in Smart Health: Efficient Policy-Hiding Attribute-Based Access Control. IEEE Internet of Things Journal. 5. 2130-2145. DOI: 10.1109/JIOT.2018.2825289.

[Zhao, 14] Zhao, Jiaqi & Wang, Lizhe & Tao, Jie & Chen, Jinjun & Weiye, Sun & Ranjan, R. & Kołodziej, Joanna & Streit, Achim & Georgakopoulos, Dimitrios. (2014). A Security Framework in G-Hadoop for Big Data Computing across Distributed Cloud Data Centres. Journal of Computer and System Sciences. 80. DOI: 10.1016/j.jcss.2014.02.006.

[Zheng, 15] Zheng, Yu. (2015). Methodologies for Cross-Domain Data Fusion: An Overview. IEEE Transactions on Big Data. 1. 1-1. DOI: 10.1109/TBDATA.2015.2465959.

[Zhou, 13] Zhou, Ruijin & Liu, Ming & Li, Tao. (2013). Characterizing the efficiency of data deduplication for big data storage management. Proceedings - 2013 IEEE International Symposium on Workload Characterization, IISWC 2013. 98-108. DOI: 10.1109/IISWC.2013.6704674.

[Zissis, 10] Zissis, Dimitris & Lekkas, Dimitrios. (2010). Addressing cloud computing security issues. Future Generation Comp. Syst.. 28. 583-592. DOI: 10.1016/j.future.2010.12.006.

Annexes

Annexe A. Code Python pour les différents modules du démonstrateur

Annexe A.1. Le module "MainFrame"

Ce module permet de lancer l'application

```
#!/usr/bin/env python
# coding: utf-8

# In[8]:


# before calling this class, make sure you have :
# - started the hadoop daemons : start-dfs.sh and start-yarn.sh
# - started the hbase service : start-hbase.sh
# - opened service thrift : hbase thrift start
# - installed all required library : happybase, recordlinkage, numpy, pandas, colorama, tkinter

# local module that treats with Data Lake
from DataLake import *

# local module that links records
from RecordsLinkage import *

# local module that calculate the accuracy
from AccuracyProcess import *

# local module that shows the chart
from Chart import *

# manage data frames : Pandas API
import pandas

# to get total time execution
import time

# manage the UI : TK-Interface API
from tkinter import filedialog
from tkinter import ttk
from tkinter import messagebox
from tkinter import *

# Main Class that launches the Big Data Accuracy Assessment API
class MainFrame(Frame):

    # Constructor
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.master = master
        self.init_window()
        self.path = None
```

```
        self.indexing_field = None
        self.file_columns = None
        # each accuracy criteria, has a weight (we need it for conflict resolution)
        self.trust_weight = 3
        self.reputation_weight = 1
        self.consistency_weight = 1
        # a spark session to get data from HDFS : one session for all the application
        self.spark_session = SparkSession.builder.getOrCreate()

    # Initilize the window
    def init_window(self):
        self.master.title("Big Data Accuracy Assessment API")
        self.main_frame = Frame(self, borderwidth=2, relief="groove", bg="#DCDCDC")
        self.main_frame.pack(side="top", fill="both", expand=True, padx=20, pady=20)
        self.pack(fill="both", expand=True)
        self.use_case_data_load_frame()

    # show the first frame : uploading use case data
    def use_case_data_load_frame(self):
        self.frame_file_to_assess = Frame(self.main_frame, borderwidth=0, highlightthickness=0,
bg="#FFFAFA")
        self.frame_file_upload = Frame(self.frame_file_to_assess, borderwidth=0, highlightthickness=0,
bg="#FFFAFA")
        self.frame_file_columns = Frame(self.frame_file_to_assess, borderwidth=0, highlightthickness=0,
bg="#FFFAFA")
        self.frame_next_to_mapping = Frame(self.frame_file_to_assess, borderwidth=0, highlightthickness=0,
bg="#FFFAFA")

        self.label_step_one = Label(self.frame_file_upload, text='Step 1 : Upload Use Case Data',
font='Helvetica 12 bold', bg="#FFFAFA")
        self.label_step_one.grid(row=0, column=0, sticky='w')

        self.label_select_file = Label(self.frame_file_upload, text='Please select a CSV file : ', bg="#FFFAFA")
        self.label_select_file.grid(row=1, column=0)

        self.button_browse = Button(self.frame_file_upload, text='Browse', command=self.choose_file)
        self.button_browse.grid(row=1, column=1, padx=50)

        self.label_csv_separator = Label(self.frame_file_upload, text='CVS Separator : ', bg="#FFFAFA")
        self.label_csv_separator.grid(row=1, column=2, padx=50)

        entry_variable = StringVar(self, value=';')
        self.entry_csv_separator = Entry(self.frame_file_upload, width=3, textvariable=entry_variable)
        self.entry_csv_separator.grid(row=1, column=3)

        self.button_reload_file = Button(self.frame_file_upload, text='Refresh',
command=self.load_local_schema)
        self.button_reload_file.grid(row=1, column=4, padx=10)

        self.frame_file_upload.pack(side="top", fill="both", expand=True, padx=10)
        self.frame_file_columns.pack(side="top", fill="both", expand=True, padx=10)
        self.frame_file_to_assess.pack(side="top", fill="both", expand=True, padx=10, pady=10)

    # local method allowing choosing the file to assess
    def choose_file(self):
        self.path = filedialog.askopenfilename(initialdir="./../datalake/opendata", title="Select csv file to
```

```
assess", filetypes=[('CSV files','.csv')])
        self.label_select_file['text'] = self.path
        self.load_local_schema()

    # load use case data schema
    def load_local_schema(self):
        # empty the frame
        for widget in self.frame_file_columns.winfo_children():
            widget.destroy()

        # load data from csv file
        dataframe = pandas.read_csv(self.path, sep=self.entry_csv_separator.get(), encoding='utf-8')
        self.file_columns = list(dataframe.columns.values)

        # fields to include in record linkage process
        self.label_all_fields = Label(self.frame_file_columns, text='Fields to include in Record Linkage Process
: ', bg="#FFFAFA")
        self.label_all_fields.grid(row =0, column = 0, sticky=N+S+W)

        i = 0
        self.check_button_list = []
        self.check_button_vars = {}
        for field in self.file_columns:
            i = i + 1
            check_var = BooleanVar()
            check_button = Checkbutton(self.frame_file_columns, text=field, variable=check_var,
bg="#FFFAFA", highlightthickness=0, bd=0)
            check_button.select()
            check_button.grid(row = i, column = 0, sticky=N+S+W)
            self.check_button_list.append(check_button)
            self.check_button_vars[field] = check_var

        # show next button
        self.button_next_to_mapping = Button(self.frame_next_to_mapping, text='Next to mapping',
command=self.go_to_mapping)
        self.button_next_to_mapping.place(relx=.5, rely=.5, anchor="center")
        self.frame_next_to_mapping.pack(side="top", fill="both", expand=True, padx=10, pady=10)

    # go to step 2 : mapping
    def go_to_mapping(self):
        # check if at least one field is checked for record linkage process
        self.use_case_fields = []
        for key, value in self.check_button_vars.items():
            if value.get():
                self.use_case_fields.append(key)
        if len(self.use_case_fields) == 0:
            messagebox.showinfo("Error", "At least, one field must be checked for Record Linkage Process")
            return

        # all is OK, let's empty the frame and start mapping frame construction
        for widget in self.main_frame.winfo_children():
            widget.destroy()
        self.use_case_data_mapping_frame()

    # show the second frame : mapping use case data with Data Lake
    def use_case_data_mapping_frame(self):
```

```
        self.frame_mapping = Frame(self.main_frame, borderwidth=0, highlightthickness=0, bg="#F0F8FF")
        self.frame_mapping.pack(side="top", fill="both", expand=True, padx=10)
        self.frame_mapping_table = Frame(self.frame_mapping, borderwidth=0, highlightthickness=0,
bg="#F0F8FF")
        self.label_step_two = Label(self.frame_mapping_table, text='Step 2 : Data Lake and Use Case Data
Mapping', font='Helvetica 12 bold', bg="#F0F8FF")
        self.label_step_two.grid(row=0, column=0, sticky='w')

        #---------------------------------- Mapping Table ----------------------------------#
        # data_lake contains all paths in Data Lake (got from hdfs_metadata)
        self.data_lake = DataLake()

        # first line : paths to data sources
        i = 1
        self.data_sources_paths_labels = {}
        for path in self.data_lake.data_sources_paths:
            path_label = Label(self.frame_mapping_table, text=path, relief=SUNKEN, borderwidth=1,
font='Helvetica 10 bold', bg="#F0F8FF").grid(row=1, column=i, ipadx=10, ipady=30)
            self.data_sources_paths_labels[path] = path_label
            i = i+1

        # first column : use case data fields
        i = 3
        self.use_case_fields_labels = {}
        for field in self.use_case_fields:
            field_label = Label(self.frame_mapping_table, text=field, relief=SUNKEN, borderwidth=1,
font='Helvetica 10 bold', bg="#F0F8FF").grid(row=i, column=0, ipady=30, sticky=W+E)
            self.use_case_fields_labels[field] = field_label
            i = i+1

        self.data_lake_fields = []
        for path in self.data_lake.data_sources_paths:
            data_source_fields = self.spark_session.read.csv(path, sep=';', encoding='utf-8',
header=True).schema.names
            self.data_lake_fields.append(data_source_fields)

        # indexing field option_menu
        i = 3
        j = 1
        self.all_option_menu_variables = {}
        for field in self.use_case_fields:
            field_option_menu_variables = []
            for data_source_fields in self.data_lake_fields:
                tk_variables = StringVar(self)
                tk_variables.set(None)
                #------ only for test : we should remove this bloc before delevrey-----#
                if i == 3 and j == 1:
                    tk_variables.set("nom_gare")
                if i == 3 and j == 2:
                    tk_variables.set("nomptar")
                if i == 3 and j == 3:
                    tk_variables.set("LIBELLE_GARE")
                if i == 4 and j == 3:
                    tk_variables.set("COMMUNE")
                if i == 5 and j == 1:
                    tk_variables.set("Geo Point")
```

```
            if i == 5 and j == 2:
                tk_variables.set("coord")
            if i == 5 and j == 3:
                tk_variables.set("coordonnees_geographiques")
            if i == 6 and j == 2:
                tk_variables.set("AnnonceSonoreProchainPassage")
            if i == 7 and j == 2:
                tk_variables.set("AnnonceVisuelleProchainPassage")
            if i == 8 and j == 2:
                tk_variables.set("AnnonceSonoreSituationsPerturbees")
            if i == 9 and j == 2:
                tk_variables.set("AnnonceVisuelleSituationsPerturbees")
            #------ only for test : we should remove the pervious bloc before delevrey-----#
            option_menu = OptionMenu(self.frame_mapping_table, tk_variables, None,
*data_source_fields).grid(row = i, column = j, sticky=W+E)
            field_option_menu_variables.append(tk_variables)
            j = j + 1
        self.all_option_menu_variables [field] = field_option_menu_variables
        j = 1
        i = i + 1

    self.frame_mapping_table.pack(side="top", fill="both", expand=True, padx=10)
    #---------------------------------- Mapping Table ----------------------------------#

    self.frame_next_to_record_linkage = Frame(self.frame_mapping, borderwidth=0, highlightthickness=0,
bg="#F0F8FF")
    self.button_next_to_record_linkage = Button(self.frame_next_to_record_linkage, text='Next to Record
Linkage', command=self.go_to_record_linkage)
    self.button_next_to_record_linkage.place(relx=.5, rely=.5, anchor="center")
    self.frame_next_to_record_linkage.pack(side="top", fill="both", expand=True, padx=10, pady=10)

  # go to step 3 : solve conflict and record linkage
  def go_to_record_linkage(self):
    self.mapping_table = {}
    for use_case_field, variables in self.all_option_menu_variables.items():
      i = 0
      mapping_row = []
      for variable in variables:
        if variable.get() != "None" :
          mapped_field = {self.data_lake.data_sources_paths[i] : variable.get()}
          mapping_row.append(mapped_field)
        i = i + 1
      self.mapping_table [use_case_field] = mapping_row
    self.launch_conflict_resolution_and_record_linkage_process()

  # launch accuracy assessment process
  def launch_conflict_resolution_and_record_linkage_process(self):
    # let's empty the frame and start conflict resolution and record linkage processes
    for widget in self.main_frame.winfo_children():
      widget.destroy()

    self.frame_record_linkage = Frame(self.main_frame, borderwidth=0, highlightthickness=0,
bg="#FFF5EE")
    self.frame_conflict_resolution = Frame(self.frame_record_linkage, borderwidth=0,
highlightthickness=0, bg="#FFF5EE")
    self.frame_next_to_accuracy = Frame(self.frame_record_linkage, borderwidth=0, highlightthickness=0,
```

```
bg="#FFF5EE")

        self.label_step_three = Label(self.frame_conflict_resolution, text='Step 3 : Mapping Conflict Resolution
and Records Linkage', font='Helvetica 12 bold', bg="#FFF5EE")
        self.label_step_three.grid(row=0, column=0, sticky='w')

        #---------- indexing fields option_menu ------------------#
        self.indexing_fields_dict = {}
        tk_variables = StringVar(self)
        tk_variables.set('Nom_Gare') # the default option
        # use case indexing field
        label_choose_indexing_field = Label(self.frame_conflict_resolution, text='Choose the indexing fields
for Use Case Data : ', bg="#FFF5EE")
        label_choose_indexing_field.grid(row = 1, column = 0, sticky=N+S+W)
        option_menu = OptionMenu(self.frame_conflict_resolution, tk_variables, *self.file_columns)
        option_menu.grid(row = 1, column = 1, sticky=N+S+W)
        self.indexing_fields_dict['use_case_data'] = tk_variables
        # Data Lake indexing field
        i = 2
        j = 0
        self.use_case_fields_labels = {}
        for path in self.data_lake.data_sources_paths:
            tk_variables = StringVar(self)
            tk_variables.set(None) # the default option
            #------ only for test : we should remove this bloc before delevrey-----#
            if j == 0:
                tk_variables.set("nom_gare")
            if j == 1:
                tk_variables.set("nomptar")
            if j == 2:
                tk_variables.set("LIBELLE_GARE")
            #------ only for test : we should remove this bloc before delevrey-----#
            label_choose_indexing_field = Label(self.frame_conflict_resolution, text='Choose the indexing fields
for ' + path + ' : ', bg="#FFF5EE")
            label_choose_indexing_field.grid(row = i, column = 0, sticky=N+S+W)
            option_menu = OptionMenu(self.frame_conflict_resolution, tk_variables, *self.data_lake_fields[j])
            option_menu.grid(row = i, column = 1, sticky=N+S+W)
            i = i + 1
            j = j + 1
            self.indexing_fields_dict[path] = tk_variables
        #---------- indexing fields option_menu ------------------#

        # show next button
        self.button_next_to_accuracy = Button(self.frame_next_to_accuracy, text='Accuracy Process',
command=self.launch_accuracy_process)
        self.button_next_to_accuracy.place(relx=.5, rely=.5, anchor="center")

        self.frame_conflict_resolution.pack(side="top", fill="both", expand=True, padx=10)
        self.frame_next_to_accuracy.pack(side="top", fill="both", expand=True, padx=10)
        self.frame_record_linkage.pack(side="top", fill="both", expand=True, padx=10)

    #----------------------------------- Accuracy Process ------------------------------#
    """
    For each value of the triplet T, R, C
        - Resolve Mapping Conflict
        - Get Linked Data Set
```

```
        - Calculate Accuracy
    """
    def launch_accuracy_process(self) :
        start_time = time.time()
        # a data frame to store all results
        results_data_frame = pandas.DataFrame(columns=['Trust' , 'Reputation', 'Consistency', 'Accuracy'])
        # loop on all possible values for T, R, C {0.00, 0.00, 0.00}, {0.00, 0.00, 0.01} ... {1.00, 1.00, 1.00} :
20400 loops
        for T in numpy.arange(0, 1.01, 0.1):
            for R in numpy.arange(0, 1.01, 0.1):
                for C in numpy.arange(0, 1.01, 0.1):
                    # resolve the mapping conflict
                    mapping_table_without_conflict = []
                    for use_case_field, mapping_row in self.mapping_table.items() :
                        mapping_dict = self.resolve_conflict(use_case_field, mapping_row, T, R, C)
                        if mapping_dict != None:
                            mapping_table_without_conflict.append(mapping_dict)
                    # build data frames pairs
                    if len(mapping_table_without_conflict) > 0:
                        data_sets_pairs =
self.get_data_sets_pairs_from_mapping_table(mapping_table_without_conflict)
                        # get linked data set
                        recordsLinkage = RecordsLinkage(data_sets_pairs, self.spark_session)
                        linked_data_set = recordsLinkage.linked_data_set
                        # get Accuracy
                        accuracyProcess = AccuracyProcess(linked_data_set)
                        A = accuracyProcess.accuracy
                    else :
                        A = None
                    print("{0:.2f}, {1:.2f}, {2:.2f} : {3}".format(T, R, C, A))
                    results_data_frame = results_data_frame.append({'Trust': T, 'Reputation': R, 'Consistency': C,
'Accuracy' : A}, ignore_index=True)

        # draw the chart
        total_time = time.time()- start_time
        print("Total time execution {0} Seconds [{1}]".format(total_time, self.format_time(total_time)))
        print("Total number of points : " + str(results_data_frame.size))
        Chart(results_data_frame, 'Trust', 'Reputation', 'Consistency', 'Accuracy')
    #------------------------------------ Accuracy Process ------------------------------#

    def resolve_conflict(self, use_case_field, mapping_row, required_trust_threshold,
required_reputation_threshold, required_consistency_threshold):
        mapping_dict = {}
        for mapped_field in mapping_row :
            data_source_path = list(mapped_field.keys())[0]
            data_source_field = list(mapped_field.values())[0]

            data_source_accuracy_trust =
float(self.data_lake.data_sources_accuracy_criterion[data_source_path]['Trust'])
            data_source_accuracy_reputation =
float(self.data_lake.data_sources_accuracy_criterion[data_source_path]['Reputation'])
            data_source_accuracy_consistency =
float(self.data_lake.data_sources_accuracy_criterion[data_source_path]['Consistency'])

            if (data_source_accuracy_trust >= required_trust_threshold and data_source_accuracy_reputation >=
required_reputation_threshold and data_source_accuracy_consistency >= required_consistency_threshold):
```

```
            global_accuracy_criterion_weight = data_source_accuracy_trust*self.trust_weight +
data_source_accuracy_reputation*self.reputation_weight +
data_source_accuracy_consistency*self.consistency_weight
            mapping_dict[global_accuracy_criterion_weight] = {"use_case_field" : use_case_field,
"data_source_path" : data_source_path, "data_source_field" : data_source_field}

        if len(mapping_dict) == 0:
            return None
        else:
            return mapping_dict[max(mapping_dict.keys())]

    def get_data_sets_pairs_from_mapping_table(self, mapping_table):
        # check if all indexing fields have been entered
        for path, tk_variable in self.indexing_fields_dict.items():
            if tk_variable.get() == 'None':
                messagebox.showinfo("Error", "Please, choose an indexing field for : " + path)
                return

        #------------------- build data set pairs ---------------------#
        data_sets_pairs = []

        # extrat data from mapping_table
        all_use_case_fields = []
        all_data_lake_fields = []
        all_data_lake_paths = []
        for row in mapping_table:
            all_use_case_fields.append(row['use_case_field'])
            all_data_lake_fields.append(row['data_source_field'])
            all_data_lake_paths.append(row['data_source_path'])
        mapped_sources_paths = list(dict.fromkeys(all_data_lake_paths))# remove duplicates

        # for each data source, build a new pair
        for mapped_source_path in mapped_sources_paths :
            use_case_fields = []
            data_lake_fields = []
            i = 0
            for path in all_data_lake_paths :
                if mapped_source_path == path :
                    use_case_fields.append(all_use_case_fields[i])
                    data_lake_fields.append(all_data_lake_fields[i])
                i = i + 1

            # first part of the pair
            use_case_attributes = {}
            indexing_field = self.indexing_fields_dict['use_case_data'].get()#tk_variable
            if indexing_field not in use_case_fields:
                use_case_fields.insert(0, indexing_field)
            use_case_attributes['path'] = self.path
            use_case_attributes['fields'] = use_case_fields
            use_case_attributes['indexing_field'] = indexing_field

            # second part of the pair
            data_lake_attributes = {}
            indexing_field = self.indexing_fields_dict[mapped_source_path].get()#tk_variable
            if indexing_field not in data_lake_fields:
                data_lake_fields.insert(0, indexing_field)
```

```
            data_lake_attributes['path'] = mapped_source_path
            data_lake_attributes['fields'] = data_lake_fields
            data_lake_attributes['indexing_field'] = indexing_field

            data_sets_pairs.append((use_case_attributes, data_lake_attributes))
        return data_sets_pairs
        #------------------ build data set pairs --------------------#

    # format total execution time (from seconds to days, hours, minutes and seconds)
    def format_time(self, secs):
        mins, secs = divmod(secs, 60)
        hours, mins = divmod(mins, 60)
        days, hours = divmod(hours, 24)
        if days > 0:
            return '%02d Days, %02d Hours, %02d Minutes, %02d Seconds' % (days, hours, mins, secs)
        if hours > 0:
            return '%02d Hours, %02d Minutes, %02d Seconds' % (hours, mins, secs)
        if mins > 0:
            return '%02d Minutes, %02d Seconds' % (mins, secs)
        if secs > 0:
            return '%02d Seconds' % (secs)

# main method : entry point to the application
if __name__=="__main__":
    root = Tk()
    root.geometry("1230x830")
    MainFrame(root)
    root.mainloop()


# In[ ]:
```

Annexe A.2. Le module "DataLake"

Ce module permet de manipuler le Data Lake (connexion à la base de données HBase, etc.)

```
#!/usr/bin/env python
# coding: utf-8

# In[10]:


# before calling this class, make sure you have :
# - started the hadoop daemons : start-all.sh (or start-dfs.sh and start-yarn.sh)
# - started the hbase service : start-hbase.sh
# - opened service thrift : hbase thrift start
# - installed required library : pip3 install happybase

import happybase

class DataLake:

    def __init__(self):
        self.connection = happybase.Connection('localhost', port=9090)
        self.table = self.connection.table('hdfs_metadata')
        self.path = 'hdfs://localhost:9000'
```

```
        self.data_sources_paths = self.path_to_all_data_sources()
        self.data_sources_accuracy_criterion = self.accuracy_criterion_of_all_data_sources()

    def path_to_all_data_sources(self) :
        data_sources_list = []
        for key, data in self.table.scan():
            data_sources_list.append(data[b'data_source:file_path'].decode())
        return data_sources_list

    def accuracy_criterion_of_all_data_sources(self) :
        accuracy_criterion_dict = {}
        for key, data in self.table.scan():
            accuracy_criterion_dict [data[b'data_source:file_path'].decode()] = {'Trust' :
data[b'accuracy_criteria:trust'].decode(), 'Reputation' : data[b'accuracy_criteria:reputation'].decode(),
'Consistency' : data[b'accuracy_criteria:consistency'].decode()}
        return accuracy_criterion_dict

# main method : entry point to the application (only for unit testing)
if __name__=="__main__":
    data_lake = DataLake()
    print(data_lake.data_sources_paths)
    for k, v in data_lake.data_sources_accuracy_criterion.items():
        print (k)
        print (v)


# In[ ]:
```

Annexe A.3. Le module "RecordsLinkage"

Ce module permet de réaliser toutes les opérations de couplage d'enregistrements et d'identification des données de référence.

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


# before calling this class, make sure you have :
# - started the hadoop daemons : start-all.sh (or start-dfs.sh and start-yarn.sh)
# - started the hbase service : start-hbase.sh
# - opened service thrift : hbase thrift start
# - installed all required library : pip3 install happybase pandas recordlinkage numpy

# import pandas (Python Data Analysis Library)
import pandas

# import recordlinkage (Python Record Linkage Toolkit : a library to link records in or between data sources)
import recordlinkage

# import all methods from local DataLake class (path_to_data_source_with_trust,
path_to_data_source_with_reputation ...)
from DataLake import *

# import numpy to manipulate matrices
```

```
import numpy

from pyspark.sql import SparkSession

class RecordsLinkage:

   def __init__(self, data_sets_pairs, sparkSession) :
      self.linked_data_set = []

      self.spark = sparkSession

      for ds1, ds2 in data_sets_pairs:
         self.linked_data_set.append(self.get_linked_data_set(ds1['path'], ds1['fields'], ds1['indexing_field'],
ds2['path'], ds2['fields'], ds2['indexing_field']))

   def get_linked_data_set(self, use_case_path, use_case_fields, use_case_indexing_field, data_lake_path,
data_lake_feilds, data_lake_indexing_feild):
      # Prepare use case Data Frame and Data Lake Data Frame
      use_case_dataframe = pandas.read_csv(use_case_path, sep=';', encoding='utf-8', usecols=use_case_fields,
dtype=str)
      data_lake_dataframe = self.spark.read.csv(data_lake_path, sep=';', encoding='utf-8',
header=True)[data_lake_feilds].toPandas()

      # Get Candidate Links by using Blocking method of RecordLinkage Library
      indexer = recordlinkage.Index()
      indexer.block(left_on=use_case_indexing_field, right_on=data_lake_indexing_feild)
      candidate_links = indexer.index(use_case_dataframe, data_lake_dataframe)

      # Get Linked Data Set
      compare = recordlinkage.Compare()
      i = 0
      for field in use_case_fields:
         compare.exact(field, data_lake_feilds[i], label=field)
         i = i + 1
      return compare.compute(candidate_links, use_case_dataframe, data_lake_dataframe)


# main method : entry point to the application (only for unit testing)
if __name__=="__main__":
   data_sets_pairs = []

   use_case_attributes = {}
   use_case_attributes['path'] = '/home/mohamed/Documents/POC/datalake/opendata/use_case_data.csv'
   use_case_attributes['fields'] = ['Nom_Gare', 'Coordonnees_Geographiques',
'Annonce_Sonore_Prochain_Passage', 'Annonce_Visuelle_Prochain_Passage',
'Annonce_Sonore_Situations_Perturbees', 'Annonce_Visuelle_Situations_Perturbees']
   use_case_attributes['indexing_field'] = 'Nom_Gare'

   data_lake_attributes = {}
   data_lake_attributes['path'] = '/user/datalake/dataset_ratp_955.csv'
   data_lake_attributes['fields'] = ['nomptar', 'coord', 'AnnonceSonoreProchainPassage',
'AnnonceVisuelleProchainPassage', 'AnnonceSonoreSituationsPerturbees',
'AnnonceVisuelleSituationsPerturbees']
   data_lake_attributes['indexing_field'] = 'nomptar'

   data_sets_pairs.append((use_case_attributes, data_lake_attributes))
```

```
    use_case_attributes = {}
    use_case_attributes['path'] = '/home/mohamed/Documents/POC/datalake/opendata/use_case_data.csv'
    use_case_attributes['fields'] = ['Nom_Gare', 'Commune']
    use_case_attributes['indexing_field'] = 'Nom_Gare'

    data_lake_attributes = {}
    data_lake_attributes['path'] = '/user/datalake/dataset_sncf_7702.csv'
    data_lake_attributes['fields'] = ['LIBELLE_GARE', 'COMMUNE']
    data_lake_attributes['indexing_field'] = 'LIBELLE_GARE'

    data_sets_pairs.append((use_case_attributes, data_lake_attributes))

    spark = SparkSession.builder.getOrCreate()
    recordsLinkage = RecordsLinkage(data_sets_pairs, spark)

    for ds in recordsLinkage.linked_data_set:
        print ('\n\n----------------------------------------')
        print(ds)
        print ('----------------------------------------\n\n')



# In[ ]:
```

Annexe A.4. Le module "AccuracyProcess"

Ce module permet de calculer l'exactitude des données en entrée.

```
#!/usr/bin/env python
# coding: utf-8

# In[39]:


import numpy

from RecordsLinkage import *

class AccuracyProcess:

    def __init__(self, linked_data_set):
        self.accuracy = self.get_accuracy(self.get_accuracy_matrices(linked_data_set))

    def get_accuracy_matrices(self, linked_data_set):
        accuracy_matrices = []

        for lds in linked_data_set:
            # Accuracy Matrix Header
            accuracy_matrix = numpy.array(['index'])
            accuracy_matrix = numpy.append(accuracy_matrix, lds.columns)
            accuracy_matrix = numpy.append(accuracy_matrix, ['record_accuracy'])

            # Accuracy Matrix construction
            for index, row in lds.iterrows():
                if row[lds.columns[0]] != 0:
```

```
                row_record_linkage = 0
                new_row = [index]
                for field in lds.columns:
                    row_record_linkage = row_record_linkage + row[field]
                    new_row.append(row[field])
                row_record_linkage = row_record_linkage / len(lds.columns)
                new_row.append(row_record_linkage)
                accuracy_matrix = numpy.vstack([accuracy_matrix, new_row])

            accuracy_matrices.append(accuracy_matrix)

        return accuracy_matrices

    def get_accuracy(self, accuracy_matrices):
        arr = accuracy_matrices[0]
        df1 = pandas.DataFrame(data=arr[1:,1:], index=[x[0] for x in arr[1:,0]], columns=arr[0,1:])
        df2 = df1.max(level=0)
        accuracy_list = df2.iloc[:,-1].mean()
        return accuracy_list

# main method : entry point to the application (only for unit testing)
if __name__=="__main__":
    data_sets_pairs = []

    use_case_attributes = {}
    use_case_attributes['path'] = '/home/mohamed/Documents/POC/datalake/opendata/use_case_data.csv'
    use_case_attributes['fields'] = ['Nom_Gare', 'Coordonnees_Geographiques',
'Annonce_Sonore_Prochain_Passage', 'Annonce_Visuelle_Prochain_Passage',
'Annonce_Sonore_Situations_Perturbees', 'Annonce_Visuelle_Situations_Perturbees']
    use_case_attributes['indexing_field'] = 'Nom_Gare'

    data_lake_attributes = {}
    data_lake_attributes['path'] = '/user/datalake/dataset_ratp_955.csv'
    data_lake_attributes['fields'] = ['nomptar', 'coord', 'AnnonceSonoreProchainPassage',
'AnnonceVisuelleProchainPassage', 'AnnonceSonoreSituationsPerturbees',
'AnnonceVisuelleSituationsPerturbees']
    data_lake_attributes['indexing_field'] = 'nomptar'

    data_sets_pairs.append((use_case_attributes, data_lake_attributes))

    use_case_attributes = {}
    use_case_attributes['path'] = '/home/mohamed/Documents/POC/datalake/opendata/use_case_data.csv'
    use_case_attributes['fields'] = ['Nom_Gare', 'Commune']
    use_case_attributes['indexing_field'] = 'Nom_Gare'

    data_lake_attributes = {}
    data_lake_attributes['path'] = '/user/datalake/dataset_sncf_7702.csv'
    data_lake_attributes['fields'] = ['LIBELLE_GARE', 'COMMUNE']
    data_lake_attributes['indexing_field'] = 'LIBELLE_GARE'

    data_sets_pairs.append((use_case_attributes, data_lake_attributes))

    spark = SparkSession.builder.getOrCreate()
    recordsLinkage = RecordsLinkage(data_sets_pairs, spark)

    accuracyProcess = AccuracyProcess(recordsLinkage.linked_data_set)
```

```
    print(accuracyProcess.accuracy)



# In[ ]:
```

Annexe A.5. Le module "Chart"

Ce module permet de représenter les résultats sur un graphique à quatre dimensions.

```
#!/usr/bin/env python
# coding: utf-8

# In[14]:


get_ipython().run_line_magic('matplotlib', 'notebook')

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

class Chart:

    # Constructor
    def __init__(self, data_frame, title, x_label, y_label, z_label, bar_label = None):
        if bar_label == None :
            self.draw_3d_plot(data_frame, title, x_label, y_label, z_label)
        else :
            self.draw_4d_plot(data_frame, title, x_label, y_label, z_label, bar_label)

    # 4D plot
    def draw_4d_plot(self, data_frame, title, x_label, y_label, z_label, bar_label):
        self.fig = plt.figure(figsize=(6,4))
        self.fig.suptitle(title, fontsize=14)
        self.ax = self.fig.add_subplot(111, projection='3d')
        self.ax.set_xlabel(x_label)
        self.ax.set_ylabel(y_label)
        self.ax.set_zlabel(z_label)
        self.plot = self.ax.scatter(data_frame[x_label], data_frame[y_label], data_frame[z_label], c =
data_frame[bar_label])
        self.color_bar = self.fig.colorbar(self.plot, ax = self.ax)
        self.color_bar.set_label(bar_label)
        plt.show()

    # 3D plot
    def draw_3d_plot(self, data_frame, title, x_label, y_label, z_label):
        self.fig = plt.figure(figsize=(4,4))
        self.fig.suptitle(title, fontsize=14)
        self.ax = self.fig.add_subplot(111, projection='3d')
        self.ax.set_xlabel(x_label)
        self.ax.set_ylabel(y_label)
        self.ax.set_zlabel(z_label)
        self.plot = self.ax.scatter(data_frame[x_label], data_frame[y_label], data_frame[z_label], c =
data_frame[z_label])
        plt.show()
```

```
# main method : entry point to the application (only for unit testing)
if __name__=="__main__":
    import numpy as np
    import pandas as pd
    from pandas import DataFrame

    results_data_frame = pd.DataFrame(columns=['Trust' , 'Reputation', 'Consistency', 'Accuracy'])
    for T in range(0, 2, 1) :
        for R in np.arange(0, 1.1, 0.05):
            for C in np.arange(0, 1.1, 0.05):
                A = (T+R+C)/4
                results_data_frame = results_data_frame.append({'Trust': T, 'Reputation': R, 'Consistency': C,
'Accuracy' : A}, ignore_index=True)

    Chart(results_data_frame, 'Plot 4D Title', 'Trust', 'Reputation', 'Consistency', 'Accuracy')

    results_data_frame = pd.DataFrame(columns=['Trust' , 'Reputation', 'Accuracy'])
    for T in range(0, 2, 1) :
        for R in np.arange(0, 1.1, 0.05):
                A = (T+R)/4
                results_data_frame = results_data_frame.append({'Trust': T, 'Reputation': R, 'Accuracy' : A},
ignore_index=True)

    Chart(results_data_frame, 'Plot 3D Title', 'Trust', 'Reputation', 'Accuracy')


# In[ ]:
```

Annexe B. Manipulation de la plateforme Hadoop

Annexe B.1. Import de données sur la plateforme Hadoop

```
# création d'un dossier « datalake » sur HDFS
o   hdfs dfs -mkdir /user
o   hdfs dfs -mkdir /user/datalake
o   hdfs dfs -ls /user/datalake
# copier des fichiers csv qui existent dans le répertoire local « opendata » vers HDFS
o   hdfs dfs -put dataset_idf_1140.csv /user/datalake/dataset_idf_1140.csv
o   hdfs dfs -put dataset_ratp_955.csv /user/datalake/dataset_ratp_955.csv
o   hdfs dfs -put dataset_sncf_7702.csv /user/datalake/dataset_sncf_7702.csv
```

Annexe B.2. Script de démarrage des services Hadoop

```
printf "\n\n--------Démarrage de l'ensemble des services Hadoop--------\n"
jps
printf "\n\n--------Démarrage de dfs : start-dfs.sh--------\n"
start-dfs.sh
printf "\n\n--------Démarrage de yarn : start-yarn.sh--------\n"
start-yarn.sh
printf "\n\n--------Démarrage de hbase : start-hbase.sh--------\n"
start-hbase.sh
printf "\n\n--------Normalement tout est prêt, vérifions avec jps--------\n"
jps
printf "\n\n--------Démarrage du service thrift de hbase : hbase thrift start--------\n"
```

```
hbase thrift start
```

Annexe B.3. Script d'arrêt des services Hadoop

```
printf "\n\n--------Arrêt des deamons------\n"
jps
printf "\n\n--------Arrêt de hbase : stop-hbase.sh------\n"
stop-hbase.sh
printf "\n\n--------Arrêt de yarn : stop-yarn.sh------\n"
stop-yarn.sh
printf "\n\n--------Arrêt de dfs : stop-dfs.sh------\n"
stop-dfs.sh
printf "\n\n------Normalement tout est arrêté, vérifions avec jps\n"
jps
```

Annexe C. Script de création de la table hdfs_metadata dans HBase

```
create 'hdfs_metadata', 'data_source', 'accuracy_criteria'

put 'hdfs_metadata', '1001', 'data_source:file_path', '/user/datalake/dataset_idf_1140.csv'
put 'hdfs_metadata', '1001', 'data_source:file_source', 'IDF'
put 'hdfs_metadata', '1001', 'accuracy_criteria:trust', '0.91'
put 'hdfs_metadata', '1001', 'accuracy_criteria:reputation', '0.42'
put 'hdfs_metadata', '1001', 'accuracy_criteria:consistency', '0.47'

put 'hdfs_metadata', '1002', 'data_source:file_path', '/user/datalake/dataset_ratp_955.csv'
put 'hdfs_metadata', '1002', 'data_source:file_source', 'RATP'
put 'hdfs_metadata', '1002', 'accuracy_criteria:trust', '0.85'
put 'hdfs_metadata', '1002', 'accuracy_criteria:reputation', '0.92'
put 'hdfs_metadata', '1002', 'accuracy_criteria:consistency', '0.75'

put 'hdfs_metadata', '1003', 'data_source:file_path', '/user/datalake/dataset_sncf_7702.csv'
put 'hdfs_metadata', '1003', 'data_source:file_source', 'SNCF'
put 'hdfs_metadata', '1003', 'accuracy_criteria:trust', '0.60'
put 'hdfs_metadata', '1003', 'accuracy_criteria:reputation', '0.65'
put 'hdfs_metadata', '1003', 'accuracy_criteria:consistency', '0.93'

scan 'hdfs_metadata'
```

Annexe D. Résultats des scénarios du cas d'étude

Annexe D.1. Résultats du Scénario 1

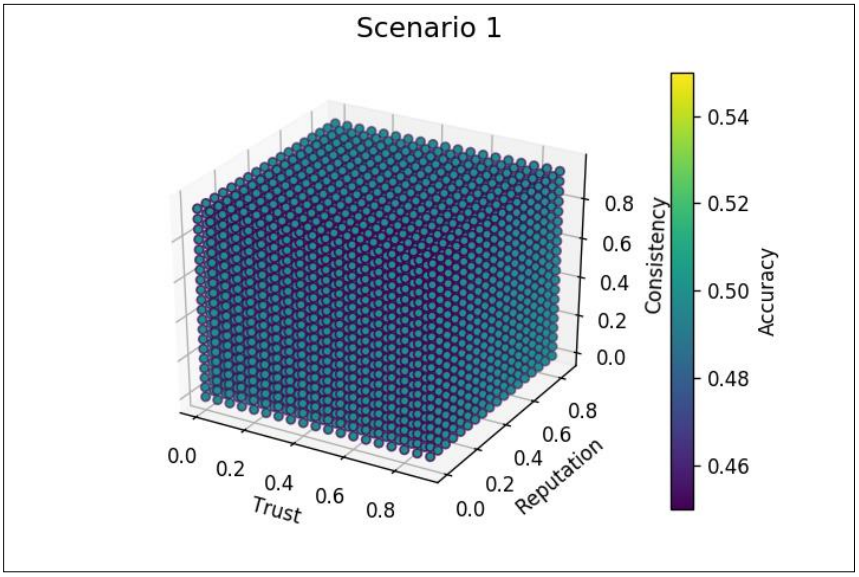

Figure D.1. Résultats de calcul de l'exactitude de données pour le Scénario 1

Table D.1.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 1

Temps d'exécution : 1963 Seconds [32 Minutes, 43 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.9	0.85	0.95	0.35	0.25	0.45	0.3	0.2	0.4
Somme pondérée	2.70			1.05			0.90		

Table D.1.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 1

Accuracy	Occurrences	Conditions
None	2421	T > 0.90 || R > 0.85 || C > 0.95
0.50	6840	T ≤ 0.90 && R ≤ 0.85 && C ≤ 0.95

Annexe D.2. Résultats du Scénario 2

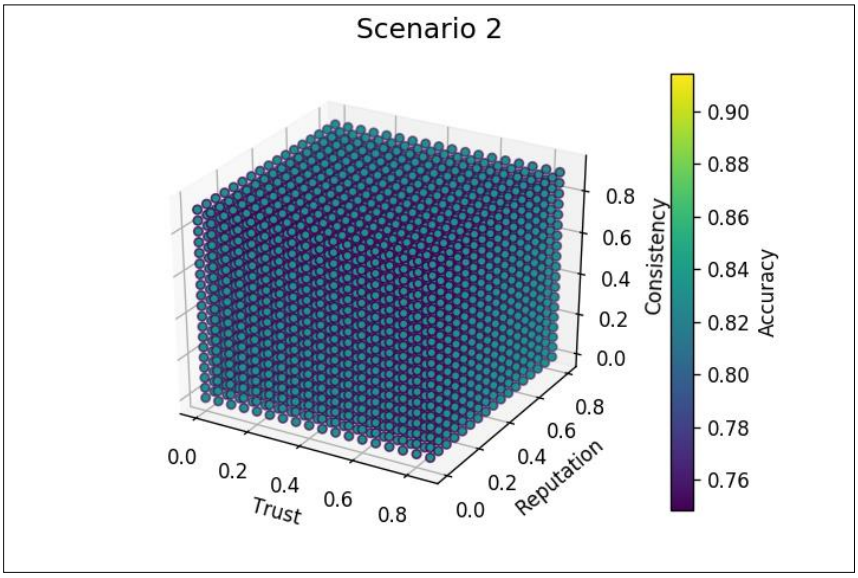

Figure D.2. Résultats de calcul de l'exactitude de données pour le Scénario 2

Table D.2.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 2

Temps d'exécution : 3090 Seconds [51 Minutes, 30 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.3	0.45	0.85	0.8	0.9	0.3	0.2	0.4
Somme pondérée	1.15			2.55			0.90		

Table D.2.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 2

Accuracy	Occurrences	Conditions
None	3447	T > 0.85 || R > 0.80 || C > 0.90
0.8314	5814	T ≤ 0.85 && R ≤ 0.80 && C ≤ 0.90

Annexe D.3. Résultats du Scénario 3

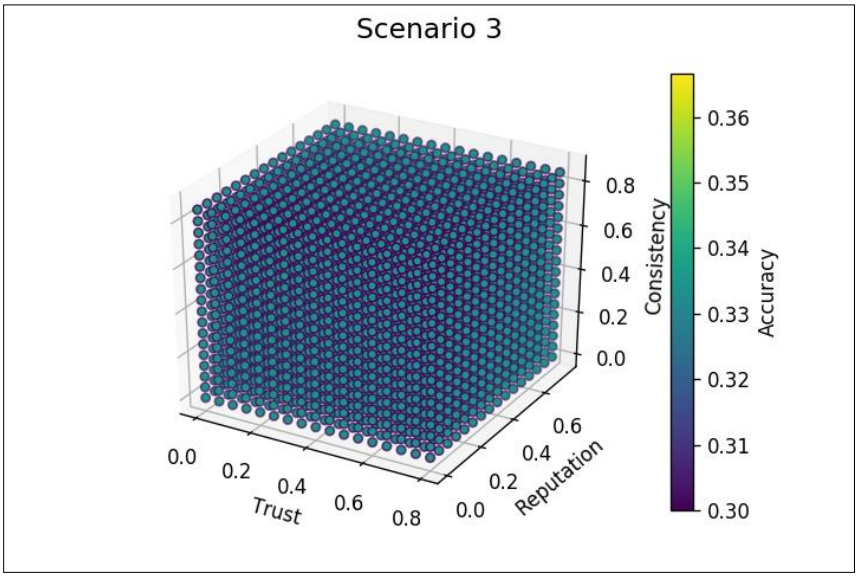

Figure D.3. Résultats de calcul de l'exactitude de données pour le Scénario 3

Table D.3.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 3

Temps d'exécution : 1501 Seconds [25 Minutes, 01 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.3	0.45	0.35	0.25	0.45	0.8	0.75	0.85
Somme pondérée	1.15			1.20			2.40		

Table D.3.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 3

Accuracy	Occurrences	Conditions
None	4365	T > 0.80 || R > 0.75 || C > 0.85
0.3333	4896	T ≤ 0.80 && R ≤ 0.75 && C ≤ 0.85

Annexe D.4. Résultats du Scénario 4

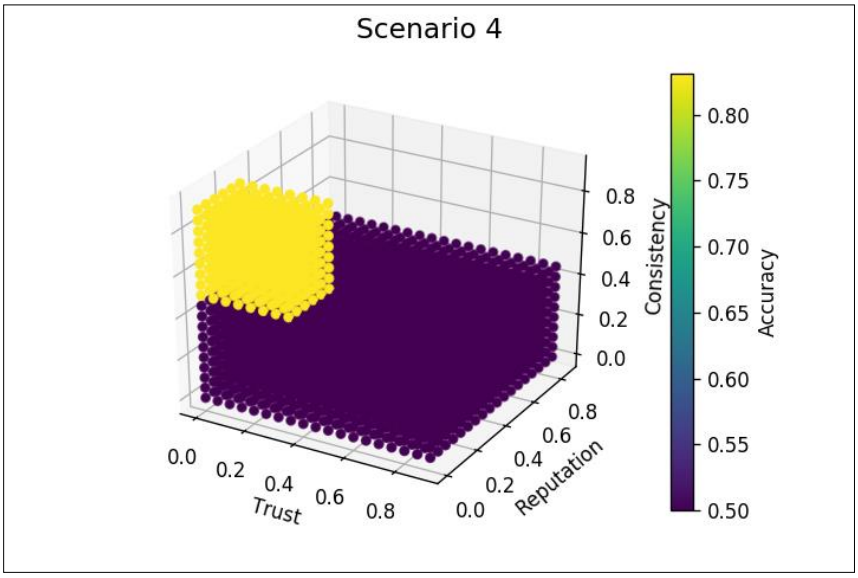

Figure D.4. Résultats de calcul de l'exactitude de données pour le Scénario 4

Table D.4.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 4

Temps d'exécution : 1266 Seconds [21 Minutes, 06 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.9	0.85	0.45	0.35	0.25	0.9	0.3	0.2	0.4
Somme pondérée	2.20			1.50			0.90		

Table D.4.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 4

Accuracy	Occurrences	Conditions
None	5409	T > 0.90 || R > 0.85 || C > 0.90 || (C > 0.45 && R > 0.25) || (C > 0.45 && T > 0.35)
0.50	3420	T ≤ 0.90 && R ≤ 0.85 && C ≤ 0.45
0.8314	432	T ≤ 0.35 && R ≤ 0.25 && 0.45 < C ≤ 0.90

Annexe D.5. Résultats du Scénario 5

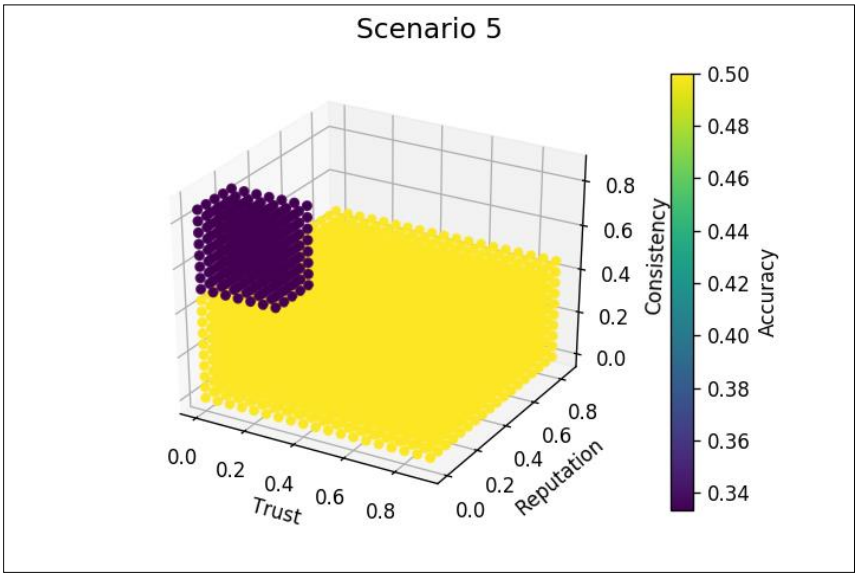

Figure D.5. Résultats de calcul de l'exactitude de données pour le Scénario 5

Table D.5.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 5

Temps d'exécution : 1142 Seconds [19 Minutes, 02 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.9	0.85	0.45	0.35	0.25	0.45	0.3	0.2	0.85
Somme pondérée	2.20			1.05			1.35		

Table D.5.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 5

Accuracy	Occurrences	Conditions
None	5561	T > 0.90 || R > 0.85 || C > 0.85 || (C > 0.45 && R > 0.20) || (C > 0.45 && T > 0.30)
0.50	3420	T ≤ 0.90 && R ≤ 0.85 && C ≤ 0.45
0.3333	280	T ≤ 0.30 && R ≤ 0.20 && 0.45 < C ≤ 0.85

Annexe D.6. Résultats du Scénario 6

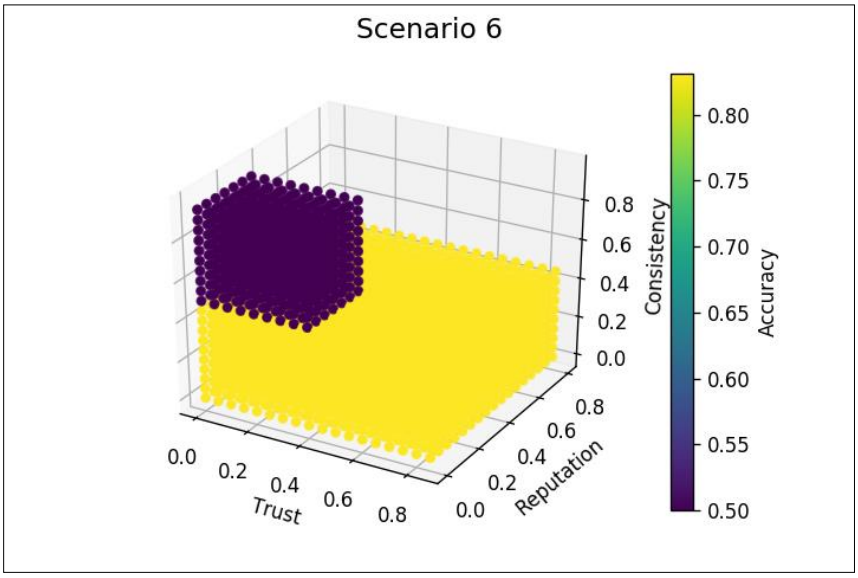

Figure D.6. Résultats de calcul de l'exactitude de données pour le Scénario 6

Table D.6.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 6

Temps d'exécution : 1785 Seconds [29 Minutes, 45 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.3	0.95	0.85	0.8	0.45	0.3	0.2	0.4
Somme pondérée	1.65			2.10			0.90		

Table D.6.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 6

Accuracy	Occurrences	Conditions
None	5571	T > 0.85 || R > 0.80 || C > 0.95 || (C > 0.45 && R > 0.30) || (C > 0.45 && T > 0.40)
0.8314	3060	T ≤ 0.85 && R ≤ 0.80 && C ≤ 0.45
0.50	630	T ≤ 0.40 && R ≤ 0.30 && 0.45 < C ≤ 0.95

Annexe D.7. Résultats du Scénario 7

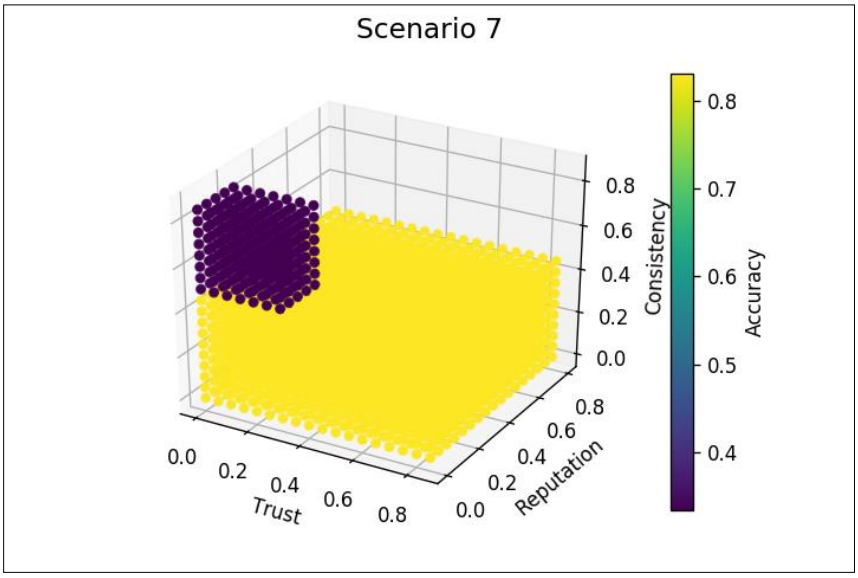

Figure D.7. Résultats de calcul de l'exactitude de données pour le Scénario 7

Table D.7.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 7

Temps d'exécution : 1647 Seconds [27 Minutes, 27 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.3	0.45	0.85	0.8	0.45	0.3	0.2	0.85
Somme pondérée	1.15			2.10			1.35		

Table D.7.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 7

Accuracy	Occurrences	Conditions
None	5921	T > 0.85 || R > 0.80 || C > 0.85 || (C > 0.45 && R > 0.20) || (C > 0.45 && T > 0.30)
0.8314	3060	T ≤ 0.85 && R ≤ 0.80 && C ≤ 0.45
0.3333	280	T ≤ 0.30 && R ≤ 0.20 && 0.45 < C ≤ 0.85

Annexe D.8. Résultats du Scénario 8

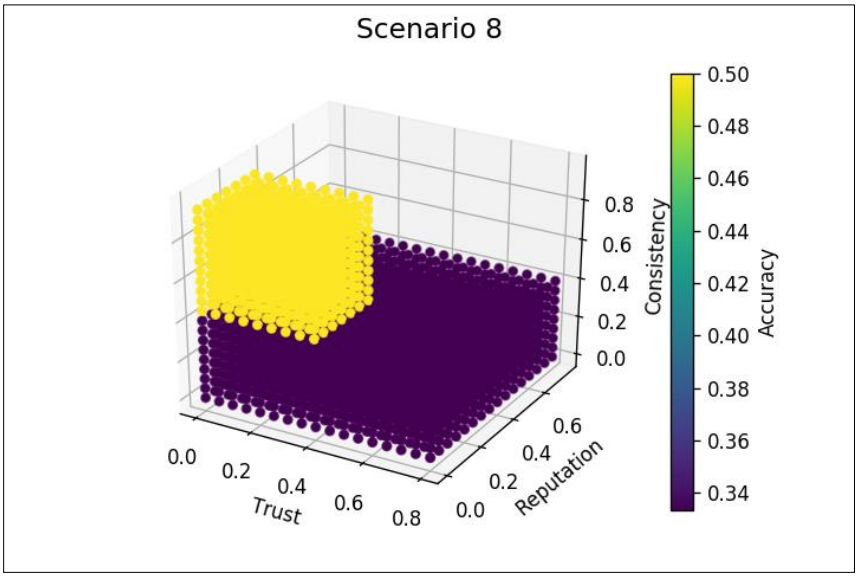

Figure D.8. Résultats de calcul de l'exactitude de données pour le Scénario 8

Table D.8.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 8

Temps d'exécution : 980 Seconds [16 Minutes, 20 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.3	0.95	0.35	0.25	0.45	0.8	0.75	0.4
Somme pondérée	1.65			1.05			1.95		

Table D.8.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 8

Accuracy	Occurrences	Conditions
None	6120	T > 0.80 || R > 0.75 || C > 0.95 || (C > 0.40 && R > 0.30) || (C > 0.40 && T > 0.40)
0.3333	2448	T ≤ 0.80 && R ≤ 0.75 && C ≤ 0.40
0.50	693	T ≤ 0.40 && R ≤ 0.30 && 0.40 < C ≤ 0.95

Annexe D.9. Résultats du Scénario 9

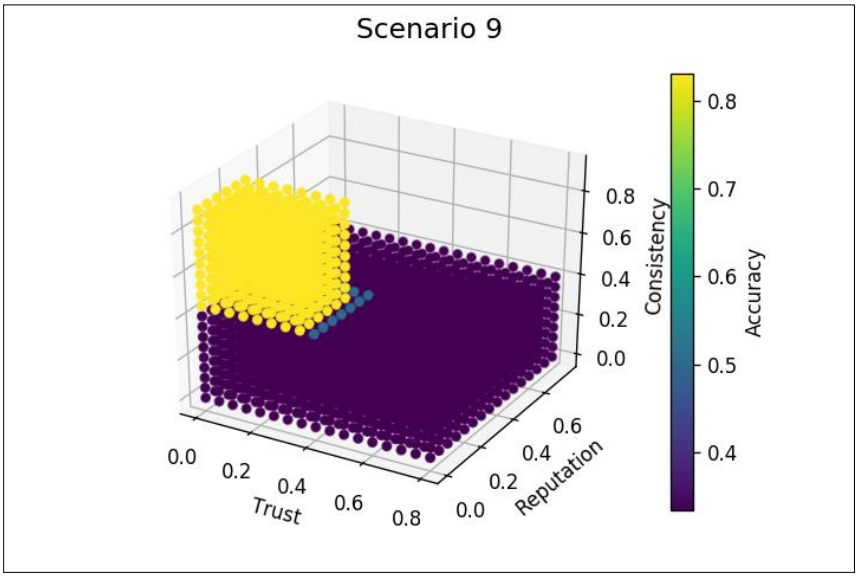

Figure D.9. Résultats de calcul de l'exactitude de données pour le Scénario 9

Table D.9.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 9

Temps d'exécution : 1046 Seconds [17 Minutes, 26 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.3	0.45	0.35	0.25	0.9	0.8	0.75	0.4
Somme pondérée	1.15			1.50			1.95		

Table D.9.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 9

Accuracy	Occurrences	Conditions
None	6318	T > 0.80 || R > 0.75 || C > 0.90 || (C > 0.40 && R > 0.25 && T > 0.40) || (C > 0.40 && T > 0.35 && R > 0.30)
0.3333	2448	T ≤ 0.80 && R ≤ 0.75 && C ≤ 0.40
0.8314	480	T ≤ 0.35 && R ≤ 0.25 && 0.40 < C ≤ 0.90
0.5	15	T ≤ 0.40 && R ≤ 0.30 && C > 0.40

Annexe D.10. Résultats du Scénario 10

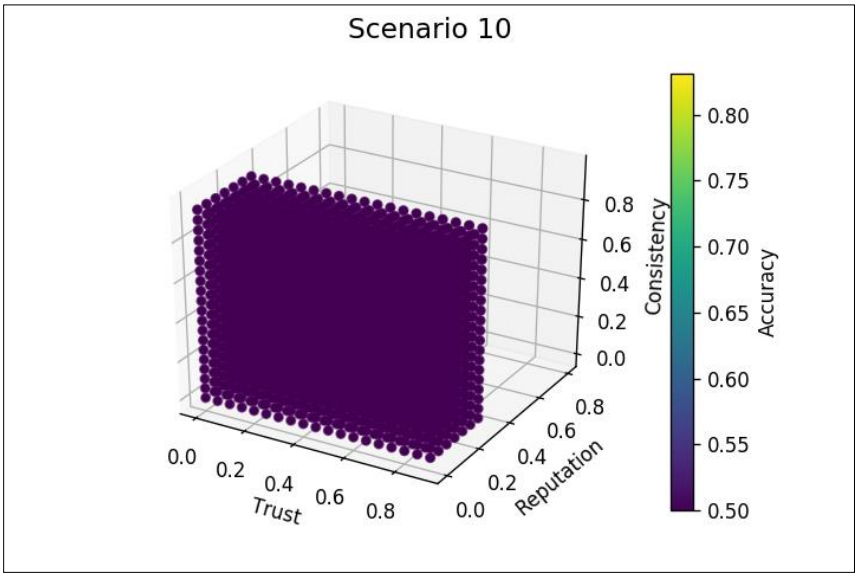

Figure D.10. Résultats de calcul de l'exactitude de données pour le Scénario 10

Table D.10.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 10

Temps d'exécution : 1234 Seconds [20 Minutes, 34 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.9	0.3	0.95	0.35	0.8	0.45	0.3	0.2	0.4
Somme pondérée	2.15			1.60			0.90		

Table D.10.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 10

Accuracy	Occurrences	Conditions
None	5801	T > 0.90 || R > 0.80 || C > 0.95 || (R > 0.30 && T > 0.35) || (R > 0.30 && C > 0.45)
0.50	2660	T ≤ 0.90 && R ≤ 0.30 && C ≤ 0.95
0.8314	800	T ≤ 0.35 && 0.3 < R ≤ 0.80 && C ≤ 0.45

Annexe D.11. Résultats du Scénario 11

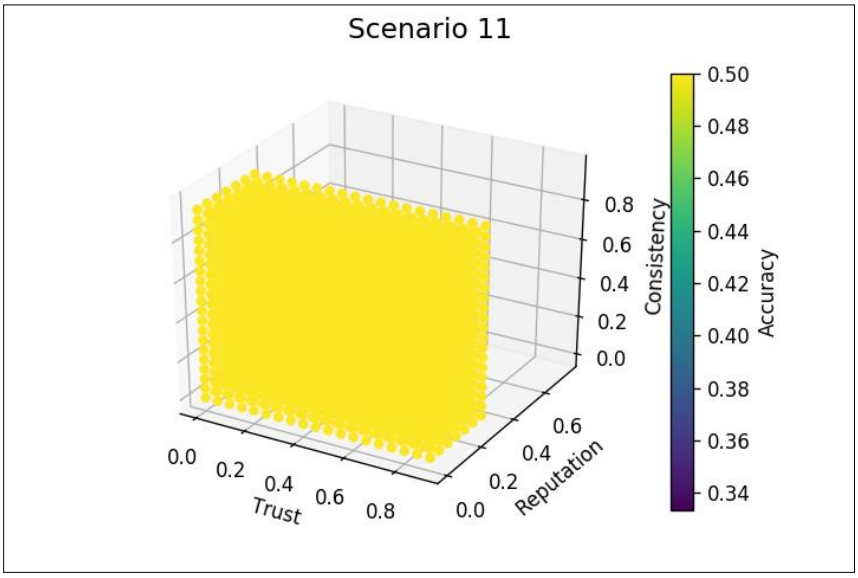

Figure D.11. Résultats de calcul de l'exactitude de données pour le Scénario 11

Table D.11.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 11

Temps d'exécution : 1051 Seconds [17 Minutes, 31 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.9	0.3	0.95	0.35	0.25	0.45	0.3	0.75	0.4
Somme pondérée	2.15			1.05			1.45		

Table D.11.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 11

Accuracy	Occurrences	Conditions
None	6034	T > 0.90 || R > 0.75 || C > 0.95 || (R > 0.30 && T > 0.30) || (R > 0.30 && C > 0.40)
0.50	2660	T ≤ 0.90 && R ≤ 0.30 && C ≤ 0.95
0.3333	567	T ≤ 0.30 && 0.30 < R ≤ 0.75 && C ≤ 0.40

Annexe D.12. Résultats du Scénario 12

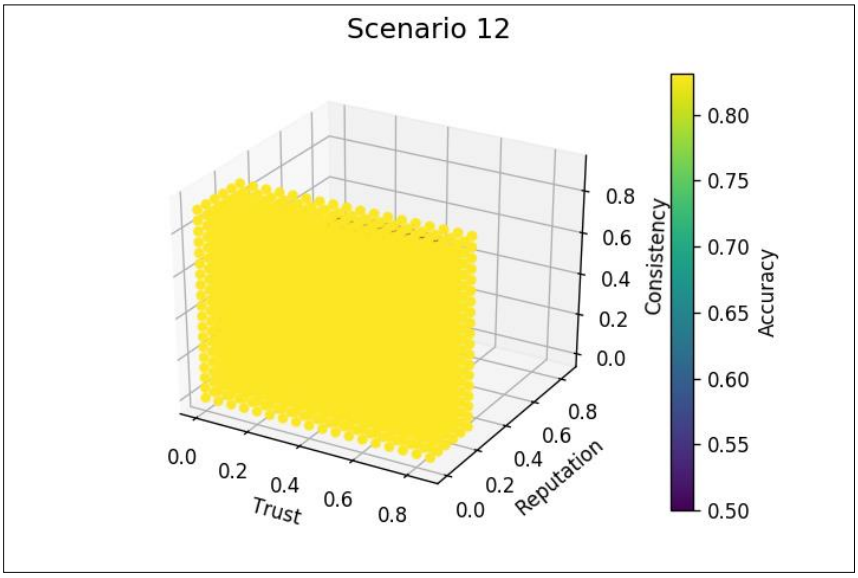

Figure D.12. Résultats de calcul de l'exactitude de données pour le Scénario 12

Table D.12.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 12

Temps d'exécution : 1348 Seconds [22 Minutes, 28 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.85	0.45	0.85	0.25	0.9	0.3	0.2	0.4
Somme pondérée	1.70			2.00			0.90		

Table D.12.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 12

Accuracy	Occurrences	Conditions
None	6129	T > 0.85 || R > 0.85 || C > 0.90 || (R > 0.25 && T > 0.40) || (R > 0.25 && C > 0.45)
0.8314	2052	T ≤ 0.85 && R ≤ 0.25 && C ≤ 0.90
0.50	1080	T ≤ 0.40 && 0.25 < R ≤ 0.85 && C ≤ 0.45

Annexe D.13. Résultats du Scénario 13

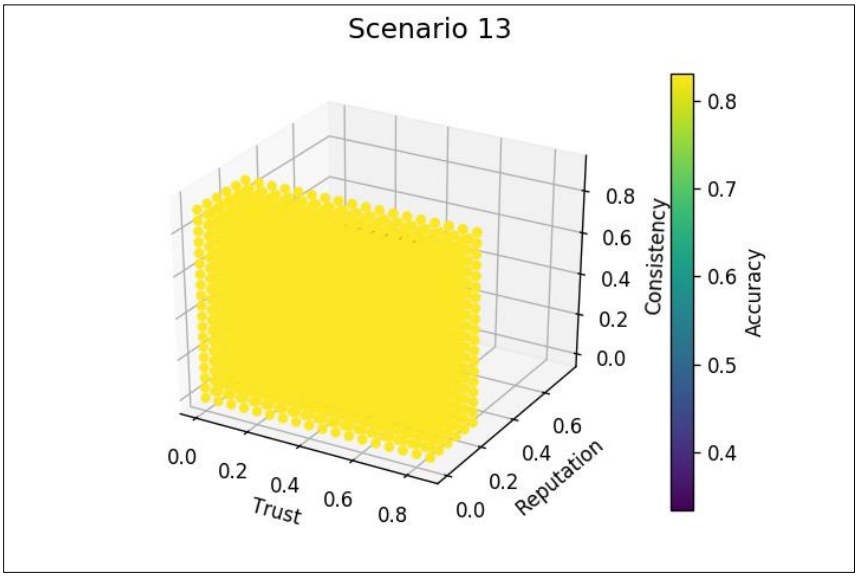

Figure D.13. Résultats de calcul de l'exactitude de données pour le Scénario 13

Table D.13.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 13

Temps d'exécution : 1257 Seconds [20 Minutes, 57 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.3	0.45	0.85	0.25	0.9	0.3	0.75	0.4
Somme pondérée	1.15			2.00			1.45		

Table D.13.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 13

Accuracy	Occurrences	Conditions
None	6552	T > 0.85 || R > 0.75 || C > 0.90 || (R > 0.25 && T > 0.30) || (R > 0.25 && C > 0.40)
0.8314	2052	T ≤ 0.85 && R ≤ 0.25 && C ≤ 0.90
0.3333	630	T ≤ 0.30 && 0.25 < R ≤ 0.75 && C ≤ 0.40
0.50	27	R = 0.30 && T ≤ 0.40 && C ≤ 0.45

Annexe D.14. Résultats du Scénario 14

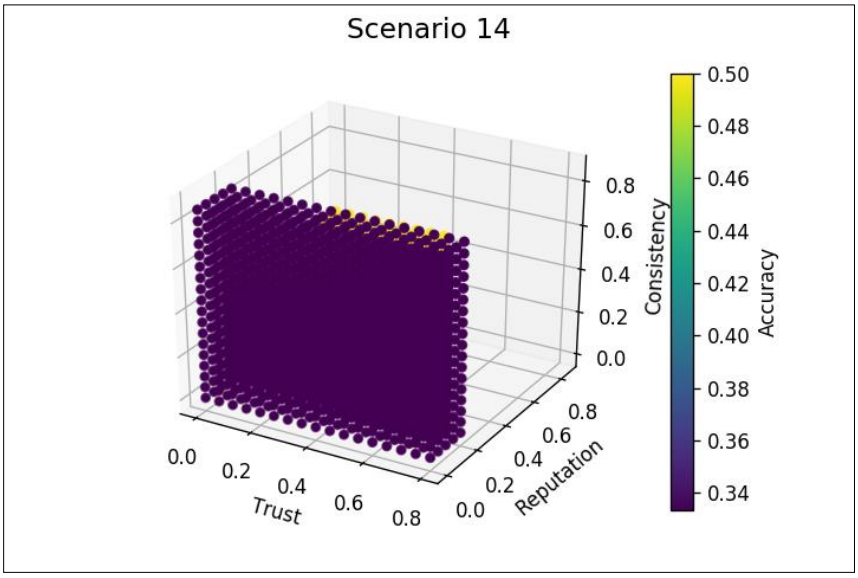

Figure D.14. Résultats de calcul de l'exactitude de données pour le Scénario 14

Table D.14.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 14

Temps d'exécution : 853 Seconds [14 Minutes, 13 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.85	0.45	0.35	0.25	0.45	0.8	0.2	0.85
Somme pondérée	1.70			1.05			1.85		

Table D.14.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 14

Accuracy	Occurrences	Conditions
None	6561	T > 0.80 || R > 0.85 || C > 0.85 || (R > 0.20 && T > 0.40) || (R > 0.20 && C > 0.45)
0.3333	1530	T ≤ 0.80 && R ≤ 0.20 && C ≤ 0.85
0.50	1170	T ≤ 0.40 && 0.20 < R ≤ 0.85 && C ≤ 0.45

Annexe D.15. Résultats du Scénario 15

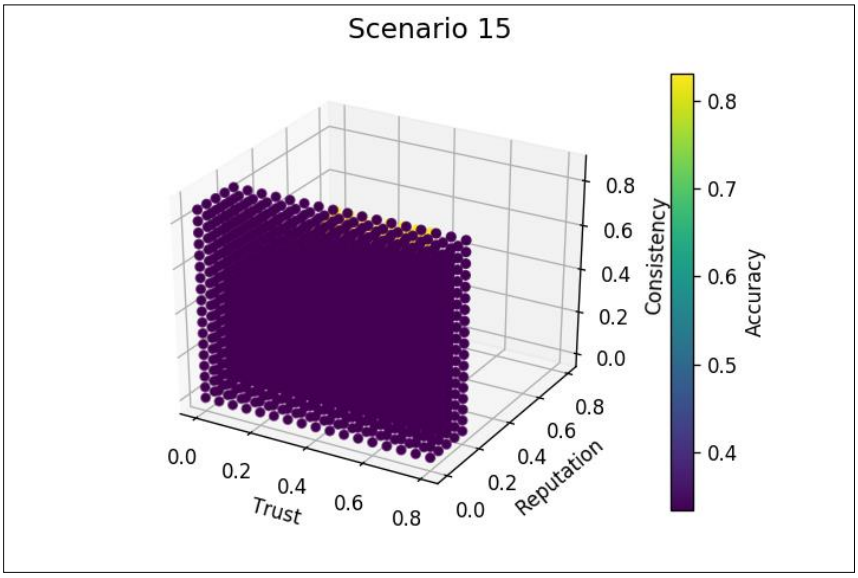

Figure D.15. Résultats de calcul de l'exactitude de données pour le Scénario 15

Table D.15.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 15

Temps d'exécution : 1040 Seconds [17 Minutes, 20 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.3	0.45	0.35	0.8	0.45	0.8	0.2	0.85
Somme pondérée	1.15			1.60			1.85		

Table D.15.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 15

Accuracy	Occurrences	Conditions
None	6751	T > 0.80 || R > 0.80 || C > 0.85 || (R > 0.20 && T > 0.35) || (R > 0.20 && C > 0.45)
0.3333	1530	T ≤ 0.80 && R ≤ 0.20 && C ≤ 0.85
0.8314	960	T ≤ 0.35 && 0.20 < R ≤ 0.80 && C ≤ 0.45
0.50	20	T = 0.40 && (R = 0.20 || R = 0.30) && C ≤ 0.45

Annexe D.16. Résultats du Scénario 16

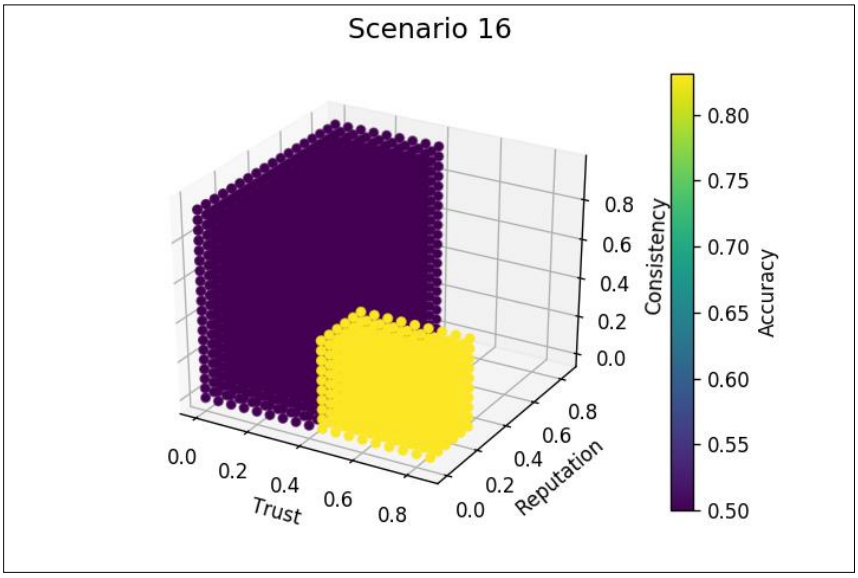

Figure D.16. Résultats de calcul de l'exactitude de données pour le Scénario 16

Table D.16.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 16

Temps d'exécution : 1246 Seconds [20 Minutes, 46 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.85	0.95	0.85	0.25	0.45	0.3	0.2	0.4
Somme pondérée	2.20			1.55			0.90		

Table D.16.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 16

Accuracy	Occurrences	Conditions
None	5481	T > 0.85 || R > 0.85 || C > 0.95 || (T > 0.40 && R > 0.25) || (T > 0.40 && C > 0.45)
0.50	3240	T ≤ 0.40 && R ≤ 0.85 && C ≤ 0.95
0.8314	540	0.40 < T ≤ 0.85 && R ≤ 0.25 && C ≤ 0.45

Annexe D.17. Résultats du Scénario 17

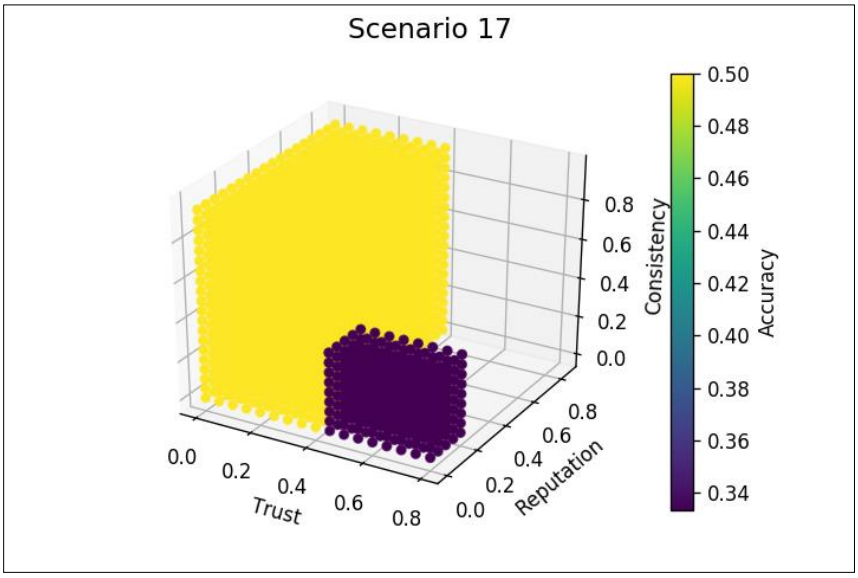

Figure D.17. Résultats de calcul de l'exactitude de données pour le Scénario 17

Table D.17.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 17

Temps d'exécution : 1142 Seconds [19 Minutes, 02 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.85	0.95	0.35	0.25	0.45	0.8	0.2	0.4
Somme pondérée	2.20			1.05			1.40		

Table D.17.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 17

Accuracy	Occurrences	Conditions
None	5661	T > 0.80 || R > 0.85 || C > 0.95 || (T > 0.40 && R > 0.20) || (T > 0.40 && C > 0.40)
0.50	3240	T ≤ 0.40 && R ≤ 0.85 && C ≤ 0.95
0.3333	360	0.40 < T ≤ 0.80 && R ≤ 0.20 && C ≤ 0.40

Annexe D.18. Résultats du Scénario 18

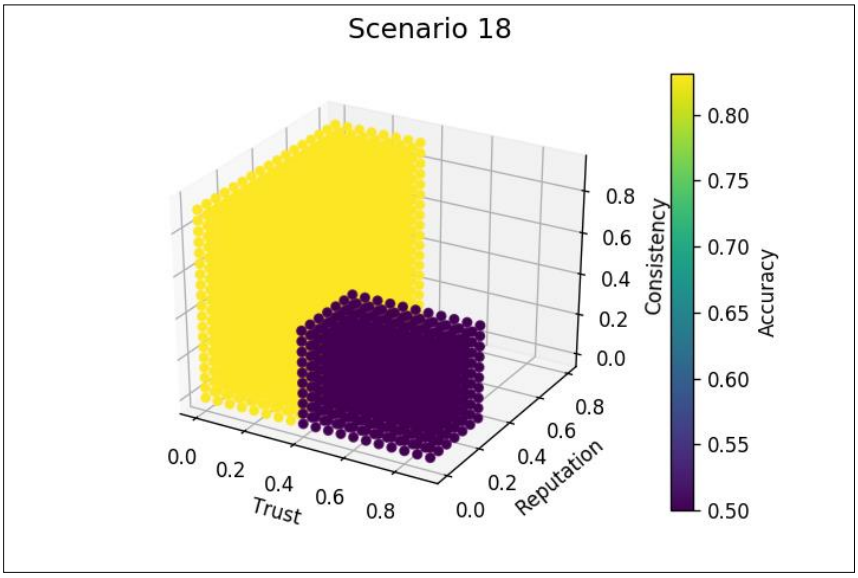

Figure D.18. Résultats de calcul de l'exactitude de données pour le Scénario 18

Table D.18.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 18

Temps d'exécution : 1550 Seconds [25 Minutes, 50 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.9	0.3	0.45	0.35	0.8	0.9	0.3	0.2	0.4
Somme pondérée	1.65			2.05			0.90		

Table D.18.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 18

Accuracy	Occurrences	Conditions
None	5907	T > 0.90 || R > 0.80 || C > 0.90 || (T > 0.35 && R > 0.30) || (T > 0.35 && C > 0.45)
0.8314	2584	T ≤ 0.35 && R ≤ 0.80 && C ≤ 0.90
0.50	770	0.35 < T ≤ 0.90 && R ≤ 0.30 && C ≤ 0.45

Annexe D.19. Résultats du Scénario 19

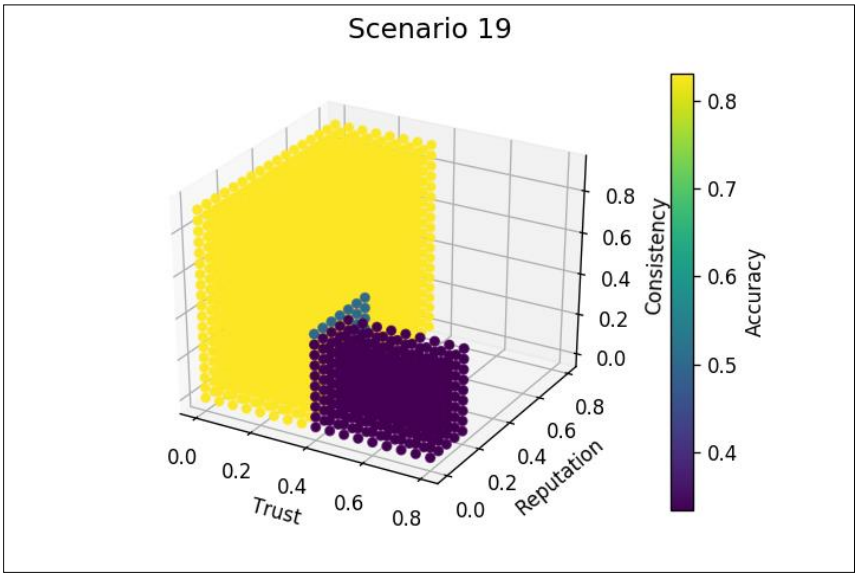

Figure D.19. Résultats de calcul de l'exactitude de données pour le Scénario 19

Table D.19.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 19

Temps d'exécution : 1477 Seconds [24 Minutes, 37 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.3	0.45	0.35	0.8	0.9	0.8	0.2	0.4
Somme pondérée	1.15			2.05			1.40		

Table D.19.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 19

Accuracy	Occurrences	Conditions
None	6247	T > 0.80 || R > 0.80 || C > 0.90 || (T > 0.35 && R > 0.20) || (T > 0.35 && C > 0.40)
0.8314	2584	T ≤ 0.35 && R ≤ 0.80 && C ≤ 0.90
0.3333	405	0.35 < T ≤ 0.80 && R ≤ 0.20 && C ≤ 0.40
0.50	25	T = 0.40 && R ≤ 0.30 && C ≤ 0.45

Annexe D.20. Résultats du Scénario 20

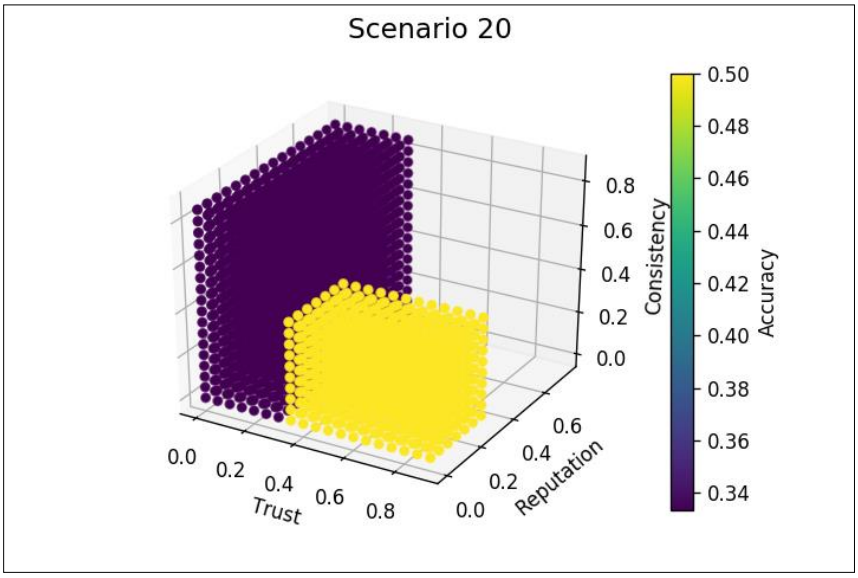

Figure D.20. Résultats de calcul de l'exactitude de données pour le Scénario 20

Table D.20.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 20

Temps d'exécution : 915 Seconds [15 Minutes, 15 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.9	0.3	0.45	0.35	0.25	0.45	0.3	0.75	0.85
Somme pondérée	1.65			1.05			1.90		

Table D.20.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 20

Accuracy	Occurrences	Conditions
None	6405	T > 0.90 || R > 0.75 || C > 0.85 || (T > 0.30 && R > 0.30) || (T > 0.30 && C > 0.45)
0.3333	2016	T ≤ 0.30 && R ≤ 0.75 && C ≤ 0.85
0.50	840	0.30 < T ≤ 0.90 && R ≤ 0.30 && C ≤ 0.45

Annexe D.21. Résultats du Scénario 21

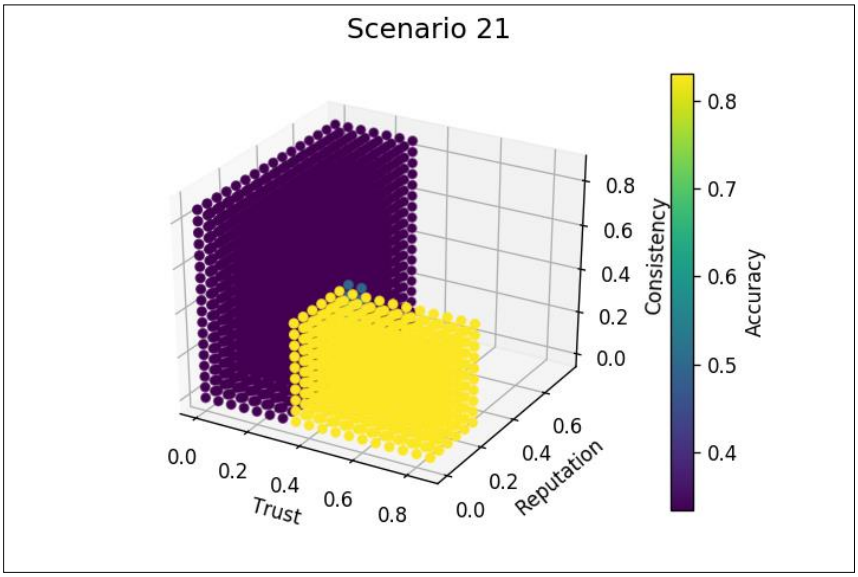

Figure D.21. Résultats de calcul de l'exactitude de données pour le Scénario 21

Table D.21.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 21

Temps d'exécution : 1034 Seconds [17 Minutes, 14 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.3	0.45	0.85	0.25	0.45	0.3	0.75	0.85
Somme pondérée	1.15			1.55			1.90		

Table D.21.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 21

Accuracy	Occurrences	Conditions
None	6565	T > 0.85 || R > 0.75 || C > 0.85 || (T > 0.30 && R > 0.25) || (T > 0.30 && C > 0.45)
0.3333	2016	T ≤ 0.30 && R ≤ 0.75 && C ≤ 0.85
0.8314	660	0.30 < T ≤ 0.85 && R ≤ 0.25 && C ≤ 0.45
0.50	20	(T = 0.35 || T = 0.45) && R = 0.30 && C ≤ 0.45

Annexe D.22. Résultats du Scénario 22

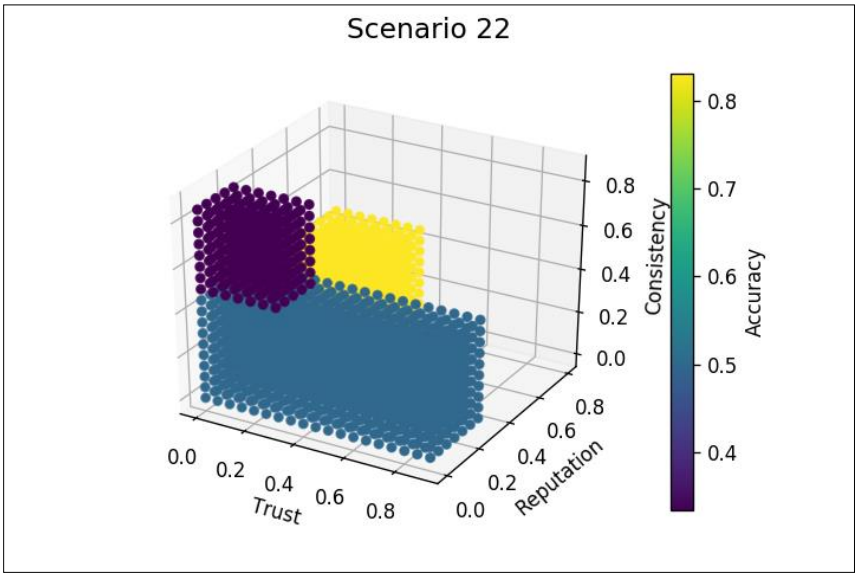

Figure D.22. Résultats de calcul de l'exactitude de données pour le Scénario 22

Table D.22.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 22

Temps d'exécution : 986 Seconds [16 Minutes, 26 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.9	0.3	0.45	0.35	0.8	0.45	0.3	0.2	0.85
Somme pondérée	1.65			1.60			1.35		

Table D.22.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 22

Accuracy	Occurrences	Conditions
None	6851	T > 0.90 || R > 0.80 || C > 0.85 || (C > 0.45 && R > 0.20) || (C > 0.45 && T > 0.30)
0.50	1330	T ≤ 0.90 && R ≤ 0.30 && C ≤ 0.45
0.8314	800	T ≤ 0.35 && 0.30 < R ≤ 0.80 && C ≤ 0.45
0.3333	280	T ≤ 0.30 && R ≤ 0.20 && 0.45 < C ≤ 0.85

Annexe D.23. Résultats du Scénario 23

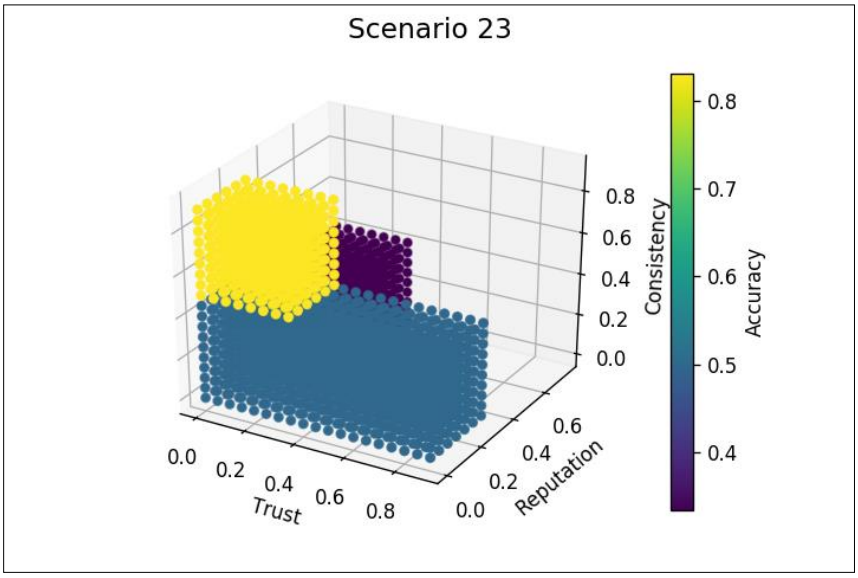

Figure D.23. Résultats de calcul de l'exactitude de données pour le Scénario 23

Table D.23.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 23

Temps d'exécution : 875 Seconds [14 Minutes, 35 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.9	0.3	0.45	0.35	0.25	0.9	0.3	0.75	0.4
Somme pondérée	1.65			1.50			1.45		

Table D.23.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 23

Accuracy	Occurrences	Conditions
None	6932	T > 0.90 || R > 0.75 || C > 0.90 || (C > 0.45 && R > 0.25) || (C > 0.45 && T > 0.35) || (R > 0.30 && T > 0.35)
0.50	1330	T ≤ 0.90 && R ≤ 0.30 && C ≤ 0.45
0.8314	432	T ≤ 0.35 && R ≤ 0.25 && C ≤ 0.90
0.3333	567	T ≤ 0.30 && R ≤ 0.75 && C ≤ 0.40

Annexe D.24. Résultats du Scénario 24

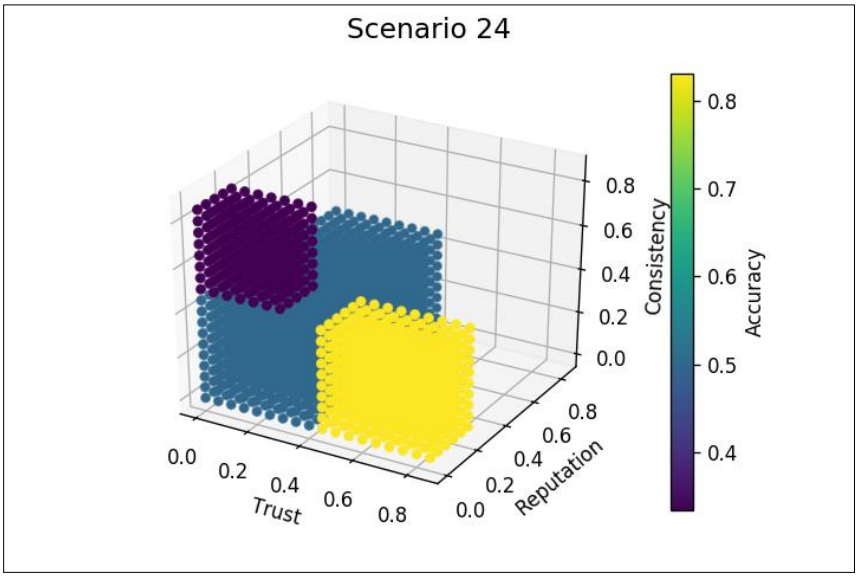

Figure D.24. Résultats de calcul de l'exactitude de données pour le Scénario 24

Table D.24.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 24

Temps d'exécution : 924 Seconds [15 Minutes, 24 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.85	0.45	0.85	0.25	0.45	0.3	0.2	0.85
Somme pondérée	1.70			1.55			1.35		

Table D.24.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 24

Accuracy	Occurrences	Conditions
None	6821	T > 0.85 || R > 0.85 || C > 0.85 || (C > 0.45 && R > 0.20) || (C > 0.45 && T > 0.30) || (R > 0.25 && T > 0.40)
0.50	1620	T ≤ 0.40 && R ≤ 0.85 && C ≤ 0.45
0.8314	540	0.40 < T ≤ 0.85 && R ≤ 0.25 && C ≤ 0.45
0.3333	280	T ≤ 0.30 && R ≤ 0.20 && 0.45 < C ≤ 0.85

Annexe D.25. Résultats du Scénario 25

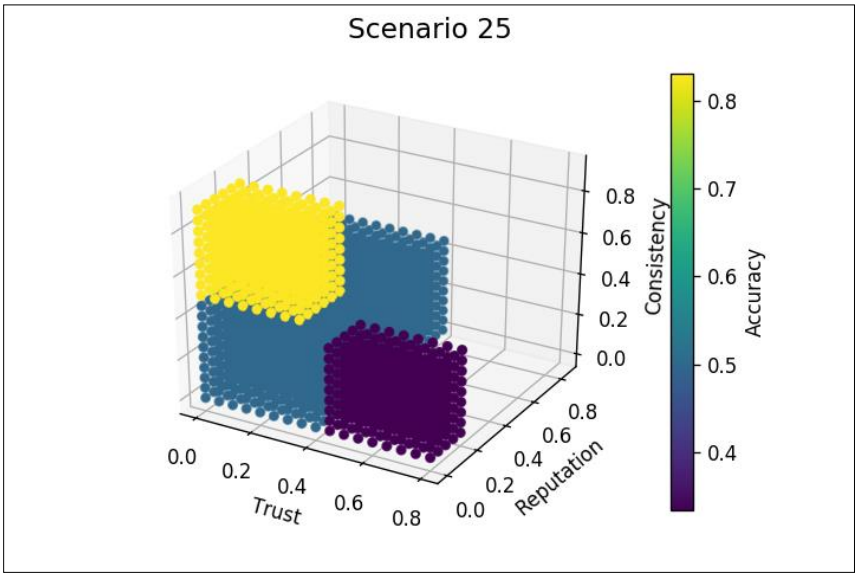

Figure D.25. Résultats de calcul de l'exactitude de données pour le Scénario 25

Table D.25.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 25

Temps d'exécution : 1047 Seconds [17 Minutes, 27 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.85	0.45	0.35	0.25	0.9	0.8	0.2	0.4
Somme pondérée	1.70			1.50			1.40		

Table D.25.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 25

Accuracy	Occurrences	Conditions
None	6849	T > 0.80 || R > 0.85 || C > 0.90 || (C > 0.45 && R > 0.25) || (C > 0.45 && T > 0.35) || (R > 0.20 && T > 0.40)
0.50	1620	T ≤ 0.40 && R ≤ 0.85 && C ≤ 0.45
0.8314	432	T ≤ 0.35 && R ≤ 0.25 && 0.45 < C ≤ 0.90
0.3333	360	0.40 < T ≤ 0.80 && R ≤ 0.20 && C ≤ 0.40

Annexe D.26. Résultats du Scénario 26

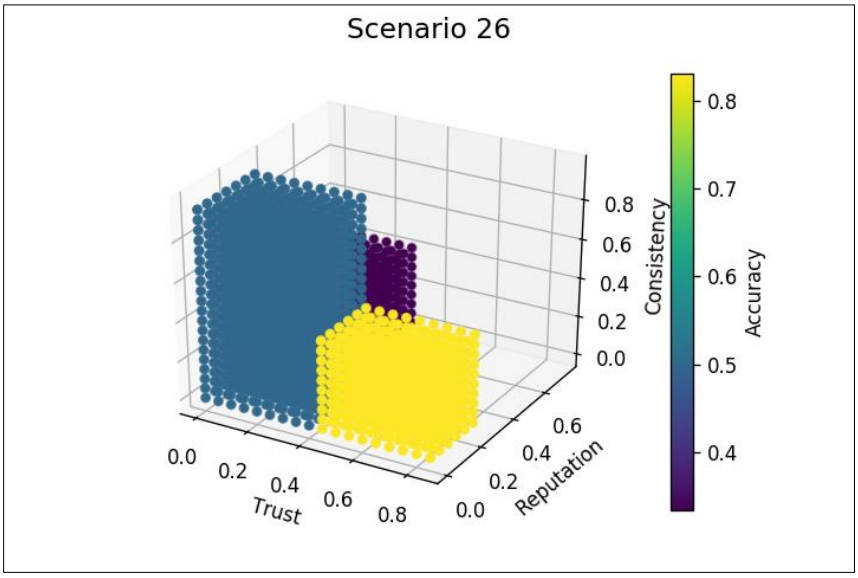

Figure D.26. Résultats de calcul de l'exactitude de données pour le Scénario 26

Table D.26.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 26

Temps d'exécution : 967 Seconds [16 Minutes, 07 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.3	0.95	0.85	0.25	0.45	0.3	0.75	0.4
Somme pondérée	1.65			1.55			1.45		

Table D.26.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 26

Accuracy	Occurrences	Conditions
None	6894	T > 0.85 || R > 0.75 || C > 0.95 || (C > 0.40 && R > 0.30) || (R > 0.30 && T > 0.30)
0.50	1260	T ≤ 0.40 && R ≤ 0.30 && C ≤ 0.95
0.8314	540	0.40 < T ≤ 0.85 && R ≤ 0.25 && C ≤ 0.45
0.3333	567	T ≤ 0.30 && 0.30 < R ≤ 0.75 && C ≤ 0.40

Annexe D.27. Résultats du Scénario 27

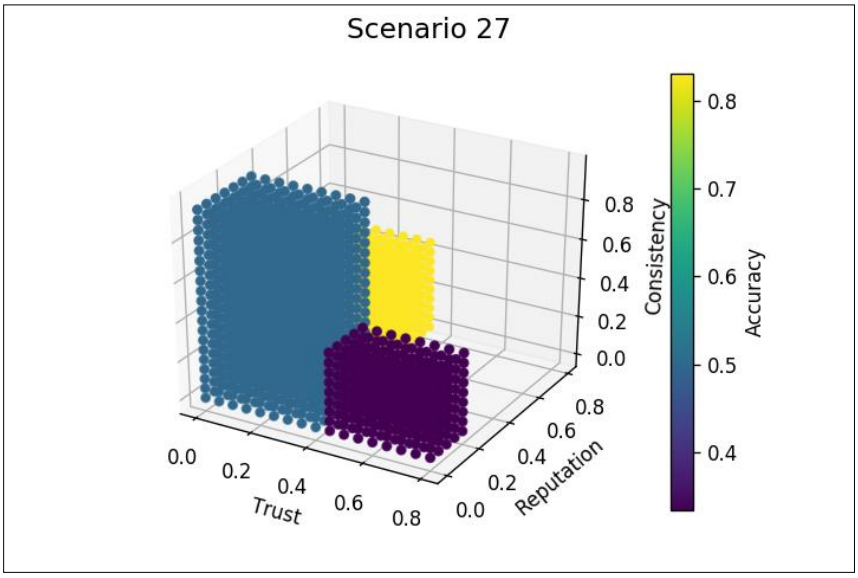

Figure D.27. Résultats de calcul de l'exactitude de données pour le Scénario 27

Table D.27.1. Temps d'exécution et somme pondérée des critères d'exactitude pour le Scénario 27

Temps d'exécution : 1051 Seconds [17 Minutes, 31 Seconds]	DS1			DS2			DS3		
	T	R	C	T	R	C	T	R	C
	0.4	0.3	0.95	0.35	0.8	0.45	0.8	0.2	0.4
Somme pondérée	1.65			1.60			1.40		

Table D.27.2. Valeurs de l'exactitude en fonction des valeurs des critères d'exactitude du Scénario 27

Accuracy	Occurrences	Conditions
None	6841	T > 0.80 || R > 0.80 || C > 0.95 || (C > 0.45 && R > 0.30) || (R > 0.30 && T > 0.35)
0.50	1260	T ≤ 0.40 && R ≤ 0.30 && C ≤ 0.95
0.8314	800	T ≤ 0.35 && 0.30 < R ≤ 0.80 && C ≤ 0.45
0.3333	360	0.40 < T ≤ 0.80 && R ≤ 0.20 && C ≤ 0.40