De la modélisation des métadonnées à la conception d'un lac de données : Application à l'habitat social

Etienne Scholly

N° d'ordre NNT : 2022LYSE2031

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON

Opérée au sein de

L'UNIVERSITÉ LUMIÈRE LYON 2

École Doctorale : ED 512 Informatique et Mathématiques

Discipline : Informatique

Soutenue publiquement le 23 mai 2022, par :

Étienne SCHOLLY

De la modélisation des métadonnées à la conception d'un lac de données.

Application à l'habitat social.

Devant le jury composé de :

Jérôme DARMONT, Professeur des universités, Université Lumière Lyon 2, Président

Anne LAURENT, Professeure des universités, Université de Montpellier, Rapporteure

Olivier TESTE, Professeur des universités, Université Toulouse 2, Rapporteur

Claudia RONCACIO, Professeure des universités, Université Grenoble Alpes, Examinatrice

Patrick MARCEL, Maître de conférences HDR, Université de Tours, Examinateur

Sabine LOUDCHER, Professeure des universités, Université Lumière Lyon 2, Co-Directrice de thèse

Cécile FAVRE, Maîtresse de conférences, Université Lumière Lyon 2, Co-Directrice de thèse

Éric FEREY, Expert, BIAL-X, Co-Directeur de thèse

Contrat de diffusion

Université Lumière Lyon 2
Ecole Doctorale Informatique et Mathématiques

THÈSE

pour obtenir le grade de
Docteur en Informatique
présentée et soutenue publiquement par

Etienne SCHOLLY

le 23 mai 2022

De la modélisation des métadonnées à la conception d'un lac de données. Application à l'habitat social

préparée au sein du laboratoire ERIC - Université Lumière Lyon 2

sous la direction de

Mme Sabine LOUDCHER
Mme Cécile FAVRE et **M. Eric FEREY**

COMPOSITION DU JURY

Mme Anne LAURENT	Rapportrice	Professeure, Université de Montpellier
M. Olivier TESTE	Rapporteur	Professeur, Université Toulouse 2
Mme Claudia RONCANCIO	Examinatrice	Professeure, Université Grenoble Alpes
M. Patrick MARCEL	Examinateur	Maitre de conférences (HDR), Université de Tours
M. Jérôme DARMONT	Examinateur	Professeur, Université Lyon 2
Mme Sabine LOUDCHER	Directrice de thèse	Professeure, Université Lyon 2
Mme Cécile FAVRE	Co-directrice de thèse	Maîtresse de conférences, Université Lyon 2
M. Eric FEREY	Invité	Président, BIAL-X

Remerciements

Tout d'abord, je tiens à remercier mes trois encadrants de thèse, Sabine, Cécile et Eric. Depuis le tout début de l'aventure, avec cette candidature tardive de ma part et une synthèse écrite alors que j'étais encore en Bretagne pour mon stage de fin d'études, puis l'entretien dans les locaux de BIAL-X qui a suivi, jusqu'à aujourd'hui, vous avez été là pour moi lors des bons moments mais aussi lors des mauvais. Vos enseignements, sur le plan scientifique, technique mais aussi humain, m'accompagneront pour toute la vie. Un immense merci de m'avoir donné l'opportunité de mener à bien cette thèse.

Je remercie les membres du jury, à savoir Anne Laurent et Olivier Teste qui ont accepté d'être les rapporteurs de ce manuscrit de thèse, mais aussi Claudia Roncancio, Patrick Marcel et Jérôme Darmont, les examinateurs de ces travaux.

Je salue bien entendu toutes les personnes de BIAL-X avec qui j'ai pu passer d'excellents moments durant 4 années, que ce soit dans le travail, en dehors, ou les deux. Merci aux « anciens » qui ont été des guides par leur professionnalisme, et merci aux « jeunes » avec qui j'ai pu passer de superbes moments. Merci à Eric et Bruno d'avoir rendu cette aventure possible. Je suis content de pouvoir dire que j'ai intégré l'entreprise à une époque où elle n'était basée qu'à Limonest, alors qu'aujourd'hui des sites à Strasbourg, à Paris et à Montpellier ont émergé. J'espère avoir réussi à apporter ma pierre à l'édifice, de quelque manière que ce soit. J'en profite pour adresser un grand merci à Baptiste M., Quentin et Mathieu qui m'ont particulièrement aidé lors du développement du « bialake ». Je fais un coucou spécialement à l'équipe BIAL-S, dont je fais désormais partie. Pour boucler la boucle, je souhaite la plus belle des thèses à Ahlame.

J'ai également été membre de l'équipe SID du laboratoire ERIC. Je salue tous les doctorants que j'ai pu croiser, en particulier Abderrazek, Antoine, Jean et Nicolas S., avec qui nous sommes montés dans le navire en même temps. Remerciements spéciaux à l'équipe des Data Lakers avec qui nous avons pu faire de belles choses : Nicolas S., Pengfei, Javier, Cécile, Sabine et Jérôme.

Cette fin de doctorat apporte le point final d'une bien longue période d'études supérieures, et de manière globale, la fin de ma scolarité au sens large. Je n'oublie pas d'où je viens, et tout le parcours effectué jusqu'ici. Merci à Mme Kilhoffer de l'école Mermoz de Schiltigheim, à M. Ferandel et Mme Mazur du collège Leclerc de Schiltigheim, à M. Stula, M. Will, M. Gutter et M. Hiegel du lycée Kléber de Strasbourg, pour tout ce

qu'ils m'ont apporté avant le baccalauréat. Merci à M. Schwarz, M. Simond, M. Prestel et Mme Kieffer de la classe prépa de Kléber, à M. Chevelu, M. Chuberre, M. Lecorvé, M. Pivert et Mme Thion de l'ENSSAT Lannion pour leurs enseignements et m'avoir aidé à me développer en tant que jeune homme.

Dédicace

Au-delà de tout ce travail de recherche, ce fut aussi une aventure humaine. Je salue les anciens de la MPSI2 et de la MP3 de Kléber, et je passe une grosse dédicace à la team Prépa (Alexandre, Anouk, Clément F., Elise, Emilie A., Emilie J., Guilhain, Gustave, Jean-Michel, Jérôme, Ludovic, Minglei, Nicolas B.) avec qui nous avons pu nous réunir plusieurs fois malgré l'éloignement géographique et les années qui passent. J'en profite pour souhaiter un prompt rétablissement à Elise, à qui j'envoie toute la force nécessaire pour vaincre cet obstacle.

Je salue tous mes camarades de l'ENSSAT, en particulier la promotion 2017, au sein de laquelle j'ai passé de formidables moments, en cours, au sport ou en soirée. C'est en particulier pendant ces années que j'ai pu apprendre l'informatique, et tous ces enseignements sont désormais gravés en moi. Salutations bien spécifiques à la Kontre'Liste qui continue à vivre tant d'années après, ainsi qu'à la team Dorelei (Adrien, Antoine, Cédric, Jean-Emmanuel, Omar), que j'espère pouvoir revoir au plus vite pour un petit DG into QG into raclette into Warp into 7v1 into Chartreuse.

Je passe une dédicace à l'élite clermontoise (Clément L., Jules, Rémi, Tanguy, Yannis) avec qui nous avons pu faire de belles promenades et balades à un rythme toujours très light. Merci aussi à Alexis de m'avoir permis d'entrer dans le monde merveilleux des vieilles BMW, et par conséquent d'être devenu une encyclopédie des pannes et problèmes des E36. Je salue les « pilotes » Nicolas W., Pascal et Wilfried, dont les petites sorties dans les Monts du Lyonnais ont malheureusement été bien trop peu nombreuses.

Salut à tous les gars que j'ai pu côtoyer au basket, que ce soit les seniors du LTB, les loisirs du TEO basket, ou les universitaires à l'ENSSAT et à Lyon 2. Merci aussi aux différents kinés qui ont eu le privilège de m'aider à reconstruire des chevilles qui en ont bien trop vu. Merci aux Bleus pour la médaille de bronze à la Coupe du monde 2019, et la médaille d'argent aux Jeux Olympiques 2020. Allez les Bulls !

Merci à Nicolas W., pour ces discussions automobiles mais aussi musique électronique. Il y a désormais un avant et un après le Goa Mix dans ma vie. J'espérais que nous pourrions un jour faire un remix de cette belle virée à Eygalières : c'est chose faite à Chaponnay, et désormais le neuvième arrondissement de Lyon compte une FiST bleue de plus (mais sans rayure).

Merci à Nicolas K., pour notre amitié qui dure dans le temps et ne faiblit pas, à l'instar d'un certain LeBron James. Notre rencontre était il y a tellement longtemps que

Boston était finaliste sortant, Luka Dončić avait 11 ans et Ronny Turiaf n'était pas encore champion NBA.

Merci à Marc, pour tes mains en or pour la mécanique et la cuisine (mais pas en même temps). C'est toujours un immense plaisir de pouvoir discuter mécanique et automobile avec toi, et aussi de te voir à l'œuvre. J'espère que je pourrai assister à une prochaine restauration dans les règles de l'art.

Merci à Clément F., pour nos mails « rouge coco », nos soirées endiablées, nos débats enflammés, nos projets visionnaires. Je lève mon verre au Dragon Rouge, et l'avenir nous dira si ce sera une coloc, un bar, une start-up ou que sais-je encore. Une chose est sûre : ce sera une légende.

Je ne serais évidemment pas là aujourd'hui sans ma famille et son soutien indéfectible. Merci à mes chers parents, même si je pense que vous dire « merci » est un bien maigre témoignage de ma gratitude pour tout ce que vous avez pu faire pour moi. Alors que mes études arrivent à leur terme, je vous souhaite la plus paisible des retraites dans votre nouvelle maison, en compagnie de Pimprenelle et Paprika.

Mes frères adorés, je vous ai couru après pendant tant d'années en espérant pouvoir être grand comme vous. C'est malin, parce qu'aujourd'hui j'ai les pieds qui dépassent du lit. Pourtant, comme disait IAM : « Petit frère veut grandir trop vite, mais il a oublié que rien ne sert de courir, petit frère ». J'espère que nous réussirons à nous retrouver régulièrement, malgré l'éloignement géographique et nos impératifs respectifs. Olivier, Baptiste, Maman, Papa, je vous aime.

Amélie, les mots manquent pour exprimer l'importance que tu as eu dans ma vie. J'aurais aimé que les choses se terminent autrement. Merci pour absolument tout.

In Memoriam
à mes professeurs, M. Schwarz et M. Simond
à ma camarade, Soukaynah
aux proches de mes proches, Germain, Françoise, Brigitte et Rahma
à mes grand-pères, Norbert et Raymond

Mamba out.

Résumé

Les années 2010 ont vu émerger le concept de lac de données (*data lake*) comme nouvelle approche pour le stockage et l'exploitation de mégadonnées (*big data*), en alternative aux entrepôts de données (*data warehouses*). Un lac de données se définit par deux propriétés principales : la variété des données qu'il est capable d'ingérer, et une approche où le schéma des données n'est défini qu'à leur interrogation (*schema-on-read*). Ces propriétés font qu'un lac de données est un système souple et adaptatif, mais nécessite en contrepartie de disposer d'un système de métadonnées efficace. En l'absence d'un schéma fixe de données, les métadonnées sont en effet essentielles pour supporter tous les usages et empêcher ainsi le lac de se transformer en marécage de données (*data swamp*), c'est-à-dire un lac de données inutilisable.

Alors que la littérature converge sur la nécessité de disposer d'un système de métadonnées efficace au sein d'un lac de données, il existe toutefois plusieurs approches pour le mettre en place. Plusieurs propositions ont déjà été formulées pour constituer un système de métadonnées, mais beaucoup de ces propositions s'avèrent être des « boites noires » difficilement réutilisables car trop peu détaillées, tandis que d'autres, plus explicites, manquent souvent de généricité pour s'adapter à des cas d'usages différents. Aussi, la mise en œuvre concrète d'un lac de données soulève son lot de problématiques, et à nouveau, plusieurs approches ont été proposées pour définir les composants majeurs d'un lac de données. Toutefois, une étude comparative des architectures fonctionnelles des lacs de données montre que ces propositions ont tendance à trop compartimenter les données du lac pour répondre à un besoin métier spécifique. Enfin, la profusion d'outils et de technologies permettant d'implémenter un lac de données vient ajouter de la confusion autour d'un concept récent et dont la définition n'est pas encore totalement consensuelle.

C'est pourquoi nous proposons dans cette thèse plusieurs contributions pour la conception, la modélisation et l'implémentation d'un lac de données et de son système de métadonnées. Nos premières contributions portent sur la modélisation des métadonnées, puisque les propositions de la littérature dans ce domaine s'avèrent manquer de généricité et ne pas être à même de prendre en charge tous types de données ou différents cas de figures importants. C'est pourquoi nous proposons un modèle de métadonnées baptisé MEDAL, que nous avons par la suite, à la lumière de travaux plus récents, fait évoluer en un métamodèle de métadonnées nommé goldMEDAL, qui se distingue des autres propositions par un niveau d'abstraction plus élevé.

En plus de la modélisation des métadonnées, nous nous sommes aussi penchés sur

la problématique de la mise en œuvre effective d'un lac de données. À ce titre, nous proposons HOUDAL, une implémentation de lac de données dédié à l'habitat social, un contexte métier qui s'inscrit dans le cadre de la thèse CIFRE, où l'entreprise BIAL-X travaille en étroite collaboration avec plusieurs bailleurs sociaux. HOUDAL se compose d'un système de métadonnées basé sur l'instanciation du métamodèle goldMEDAL, ainsi que d'une interface web avec laquelle l'utilisateur interagit pour accéder au lac de données, i.e. aux données et aux métadonnées. Pour aider l'utilisateur lors de son utilisation de HOUDAL, nous proposons aussi QSTR, un assistant à la création de métadonnées pour les données structurées. QSTR se situe dans la couche d'ingestion de HOUDAL et aide l'utilisateur à décrire de manière efficiente les données structurées, en particulier dans l'optique de décrire des évolutions de schéma qui pourraient subvenir lors de l'ajout de nouvelles occurrences de données au sein du lac.

Mots-clés : lacs de données, modélisation des métadonnées, implémentation, assistant

Summary

Title : "From metadata modeling to the conception of a data lake. Application to public housing"

The 2010s saw the emergence of the data lake concept as a new approach for storing and exploiting big data, as an alternative to data warehouses. A data lake is defined by two main properties : the diversity of data it is capable of ingesting, and an approach where the data schema is defined only when querying it (*schema-on-read*). These properties make a data lake a flexible and adaptive system, but require an efficient metadata system. In the absence of a fixed data schema, metadata are indeed essential to support all uses and thus prevent the lake from turning into a data swamp, i.e. an unusable data lake.

While the literature converges on the need for an efficient metadata system within a data lake, there are however several approaches to establish it. Several proposals have already been formulated to build a metadata system, but many of these proposals turn out to be "black boxes" that are difficult to reuse because they are not detailed enough, while others, which are more explicit, often lack the genericity to adapt to different use cases. Also, the concrete implementation of a data lake raises its share of problems, and again, several approaches have been proposed to define the major components of a data lake. However, a comparative study of the functional architectures of data lakes shows that these proposals tend to overly compartmentalize data in the lake to meet a specific business need. Finally, the profusion of tools and technologies allowing the implementation of a data lake adds to the confusion around a recent concept whose definition is not yet totally consensual.

Therefore, we propose in this thesis several contributions to the conception, modeling and implementation of a data lake and its metadata system. Our first contributions concern metadata modeling, since the proposals in the literature in this area turn out to lack genericity and are not able to support any type of data or different important situations. This is why we propose a metadata model named MEDAL, which we have subsequently, in the light of more recent work, evolved into a metadata metamodel named goldMEDAL, which differs from other proposals by its higher level of abstraction.

In addition to the modeling of metadata, we also addressed the issue of the actual implementation of a data lake. For this purpose, we propose HOUDAL, a data lake implementation dedicated to social housing, a business context that falls within the fra-

mework of the CIFRE thesis, where the company BIAL-X works in close collaboration with several social landlords. HOUDAL is composed of a metadata system based on the instantiation of the goldMEDAL metamodel, as well as a web interface allowing the user to access the data lake, i.e. data and metadata. To help the user when using HOUDAL, we also propose QSTR, a metadata creation wizard for periodically generated structured data. QSTR is located in the ingestion layer of HOUDAL and helps the user to efficiently describe structured data, especially in order to describe schema evolutions that may occur when adding new data occurrences to the lake.

Keywords : data lakes, metadata modelization, implementation, wizard

"Those times when you get up early and you work hard, those times when you stay up late and you work hard, those times when you don't feel like working, you're too tired, you don't want to push yourself, but you do it anyway. That is actually the dream. That's the dream. It's not the destination, it's the journey. And if you guys can understand that, then what you'll see happen is you won't accomplish your dreams, your dreams won't come true ; something greater will."

Kobe Bryant

Chapitre 1

Introduction générale

Sommaire

"I know what it's like to lose. To feel so desperately that you're right, yet to fail nonetheless. It's frightening, turns the legs to jelly. But I ask you, to what end ? Dread it. Run from it. Destiny arrives all the same. Or should I say...I AM."

Thanos, *Avengers : Infinity War (2018)*

1.1 Contexte industriel de BIAL-X

BIAL-X et l'habitat social

La thèse de doctorat s'inscrit dans le cadre d'une convention CIFRE entre le laboratoire ERIC [1] et la société BIAL-X [2]. Historiquement, BIAL-X est un cabinet d'expertise en informatique décisionnelle (*business intelligence*), où les consultants de la société accompagnent des clients dans des projets ou en régie pour la mise en place et la maintenance d'un système d'information décisionnel. Il y a maintenant plusieurs années, BIAL-X a effectué un virage stratégique pour élargir son domaine d'expertise à la science des données (*data science*), et s'intéresse donc à l'extraction de connaissances à partir des données au sens large. Cet élargissement des activités de BIAL-X s'est accompagné de la création d'une seconde entité nommée BIAL-R [3], les deux sociétés étant réunies au sein de la *Data Intelligence Agency* (DIA). Les travaux menés au cours de cette thèse de doctorat s'inscrivent dans le développement de la DIA, pour incorporer aux offres commerciales des entités BIAL de nouvelles solutions innovantes, avec un intérêt scientifique certain.

La DIA travaille avec de nombreux clients issus de secteurs économiques différents. Parmi eux se trouvent les bailleurs sociaux, qui sont des organismes louant des logements à un loyer modéré à des ménages ayant peu de revenus. Ce sont des clients historiques et importants pour l'entreprise, dont certains ont fait confiance à BIAL-X pour la création puis la maintenance de leur système d'information décisionnel. Les données des systèmes sources sont extraites et traitées pour être ensuite intégrées au sein d'un entrepôt de données, ce dernier servant de base à des tableaux de bord utilisés par les bailleurs pour piloter leurs activités. À l'aide d'un tel système, ils peuvent aisément surveiller les évolutions concernant le patrimoine (achats, ventes, travaux, etc.), ou encore avoir des informations sur les locataires (entrées, départs, sollicitations, litiges, etc.). Dans cette lignée, BIAL-X propose dans son offre commerciale la *BialBOX*, un écosystème décisionnel complet qui se connecte aux données sources d'un client et s'intègre parfaitement à son système d'information. La *BialBOX* se compose de trois grandes parties : l'alimentation qui vise à extraire et préparer les données, le socle de stockage des données et la restitution pour présenter les données.

De même, des bailleurs ont placé leurs espoirs en la DIA pour mener des analyses poussées sur leurs données afin d'en extraire des connaissances dont ils ne disposaient pas forcément auparavant. Ceci est rendu possible par l'utilisation de méthodes de sciences de données, et plus spécifiquement d'intelligence artificielle, rendant possible des analyses prédictives. À titre d'exemple, l'entreprise a pu fournir à des bailleurs des prédictions concernant les locataires ayant un profil susceptible d'avoir un impayé prochainement, ou bien de prévoir avant un état des lieux sortant si des travaux de rénovation du logement allaient être nécessaires. La DIA propose une seconde offre commerciale nommée *BialVoG* (pour *Value of Gain*), dont l'objectif est d'extraire des informations pertinentes

1. `https://eric.msh-lse.fr/`
2. `https://www.bial-x.com/`
3. `https://www.bial-r.com/`

et nouvelles à partir des données du client, et ce dans un délai très court en utilisant la méthodologie CRISP/DM (*Cross-Industry Standard Process for Data Mining*) [Wirth and Hipp, 2000]. La *BialVoG* vise à objectiver et démontrer des gains atteignables pour le client.

L'émergence des mégadonnées apporte évidemment de nombreuses nouvelles opportunités aux bailleurs sociaux. Il devient d'une part possible d'analyser de nouvelles données, structurées ou non, comme par exemple les appels téléphoniques entre un bailleur et un locataire à propos d'un litige, ou des échanges sur les réseaux sociaux. De même, les données ouvertes (*open data*) peuvent apporter aux bailleurs des informations qu'ils ne possèdent pas dans leurs données, à travers par exemple des données géographiques. C'est justement l'objet d'une autre offre commerciale de l'entreprise, appelée *BialGEO*, qui vient compléter l'offre *BialVoG* à partir de données géographiques. Ainsi, aux concepts de la *BialVoG* s'ajoutent les techniques de géocodage, calcul de proximité, référentiels SIG (système d'information géographique), outils d'analyse spatiale ou encore construction de géométries personnalisées.

De nouveaux enjeux pour les bailleurs sociaux

Les bailleurs sociaux sont capables d'extraire des connaissances à partir de données à l'aide de méthodes issues de l'informatique décisionnelle ou de la science des données. La révolution des mégadonnées ouvrant encore plus le champ des possibles, les bailleurs sociaux ont désormais la volonté de gagner en agilité pour se laisser la possibilité de mener de nouvelles analyses sur leurs données, mais aussi les données ouvertes du web. En plus des analyses « traditionnelles » qui sont utilisées couramment par les bailleurs (l'informatique décisionnelle, par exemple), leur objectif est de pouvoir essayer de répondre à de nouveaux besoins d'analyse, qui sont souvent des preuves de concept.

Prenons un exemple concret. Les bailleurs disposent généralement d'un grand nombre d'informations sur leurs logements (superficie, nombre de pièces, type de chauffage, date de construction, etc.), mais ont en revanche peu voire pas d'information sur l'environnement dudit logement (transports, services publics, sécurité, emploi, etc.). Ces dernières informations sont néanmoins de plus en plus mises à disposition sur le web en tant que données ouvertes. Croiser ces données externes concernant l'environnement d'un logement avec les données internes du logement permettrait de répondre à des questions auxquelles il est difficile de trouver des réponses uniquement avec les données des bailleurs. Par exemple, un logement refait à neuf, spacieux et lumineux mais restant vacant peut difficilement s'expliquer, sauf à avoir la connaissance qu'il est situé dans un quartier isolé, avec peu de transports ou une forte insécurité.

Pour généraliser, les bailleurs voudraient mener diverses analyses sur tous types de données, et ce lorsque le besoin se fait sentir. À l'heure actuelle, la DIA est capable d'aider les bailleurs sociaux à répondre à ces nouveaux besoins ponctuels, mais elle rencontre néanmoins plusieurs problèmes. Tout d'abord, ces nouvelles analyses de données sont presque systématiquement menées comme des preuves de concept, c'est-à-dire de manière *ad hoc*, isolées des autres exploitations de données du bailleur déjà en place.

Il est très difficile de mettre en place une gouvernance centralisée des données : chaque nouveau besoin génère des documents, tels que des dictionnaires et/ou des catalogues de données, mais qui rarement reliés aux documents déjà existants chez le bailleur. Lorsque les preuves de concept se multiplient, ces réponses à des besoins ponctuels sont difficilement réutilisables, et finissent par tomber dans l'oubli. Alors, le bailleur ne peut pas capitaliser sur ces preuves de concept, alors qu'elles représentent un investissement humain et financier conséquent.

Pour toutes ces raisons, l'entreprise souhaite donc accompagner ses clients (qui sont, dans l'exemple retenu pour la thèse, des bailleurs sociaux) qui souhaitent pouvoir convenablement tirer profit de l'exploitation des mégadonnées. Toutefois, exploiter des données de tous types est une véritable problématique scientifique, avec plusieurs tenants et aboutissants, et y apporter une (ou des) solution(s) est tout sauf trivial. C'est justement cette problématique intéressante mais complexe qui a motivé les travaux de recherche menés lors de la thèse. Ainsi, de cette problématique métier découlent des problématiques de recherche, que nous présentons ci-dessous.

1.2 Problématique de recherche et objectifs de la thèse

Mener tous types d'analyses sur des données variées, potentiellement des mégadonnées, s'inscrit dans la révolution du *big data* et ses problèmes associés [Chen et al., 2014]. Les mégadonnées sont usuellement caractérisées par leur Volume, leur Variété et leur Vélocité, c'est-à-dire les « 3 V » [Assunção et al., 2015] désignant le fait qu'elles peuvent être massives, de formats variés, ou encore créées en temps réel. Ces caractéristiques les rendent difficiles à entreposer et exploiter avec les outils traditionnels, ce qui de fait pose de nombreux problèmes à résoudre [Gandomi and Haider, 2015]. Toutefois, trouver des solutions à ces verrous scientifiques mènerait alors à de très nombreuses opportunités [Beheshti et al., 2018, Sivarajah et al., 2017].

Les approches traditionnelles pour le stockage et l'analyse de données, comme par exemple les entrepôts de données [Inmon, 1996, Kimball, 2008, Watson and Wixom, 2007], sont peu adaptés pour gérer les mégadonnées. De plus, le principe de fonctionnement d'un entrepôt où le schéma des données doit être défini à l'enregistrement (*schema-on-write*) montre aussi ses limites face à des données qui peuvent être très hétérogènes [Khine and Wang, 2017, LaPlante and Sharma, 2016].

Il existe ainsi un réel besoin de savoir comment stocker, organiser, interroger et analyser des mégadonnées [Chen et al., 2012]. Comme piste de solution, Dixon propose le concept de lac de données (*data lake*), qui se définit comme un vaste dépôt de données hétérogènes et à partir duquel tous types d'analyses peuvent être réalisées [Dixon, 2010]. Un lac de données repose sur deux propriétés principales : la variété des données qu'il peut contenir, et le principe de fonctionnement *schema-on-read*, c'est-à-dire que le schéma des données est défini seulement à l'interrogation [Miloslavskaya and Tolstoy, 2016].

Toutefois, le concept de lac de données est relativement récent et de ce fait, plusieurs problématiques de recherche subsistent. D'une part, puisque les données sont enregistrées sans schéma prédéfini, un lac de données doit nécessairement disposer d'un système de

métadonnées efficace, qui permet de s'y retrouver au sein du lac malgré la diversité des données qu'il peut contenir. Sans métadonnées, le lac de données est inutilisable et devient un marécage de données (*data swamp*) [Alrehamy and Walker, 2015, Suriarachchi and Plale, 2016, Hai et al., 2016]. Plus particulièrement, la question d'une modélisation « générique » des métadonnées du lac est une problématique sur laquelle des équipes de recherche se penchent [Diamantini et al., 2018, Hellerstein et al., 2017, Quix et al., 2016].

Un second problème de recherche se pose lors de la mise en œuvre concrète d'un lac de données. La définition de l'architecture fonctionnelle d'un lac de données, i.e. les principaux éléments qui le composent, est un sujet qui fait débat [John and Misra, 2017, LaPlante and Sharma, 2016]. Ceci est lié au fait que la définition du concept de lac de données n'a été stabilisée que récemment, et donc que les auteurs définissaient des architectures de lac selon leur propre définition [Madera and Laurent, 2016]. En outre, bien que le concept de lac de données ne soit pas très vieux, il existe déjà un grand nombres de technologies permettant d'en implémenter. Le terme de lac de données est d'ailleurs souvent associé uniquement à la technologie Apache Hadoop, ce qui vient amener de la confusion supplémentaire [Fang, 2015, O'Leary, 2014]. Lors de l'implémentation, il faut aussi être vigilant à rendre le lac facile à utiliser.

À la lumière de ces éléments, notre problématique de recherche porte donc sur **la conception, la modélisation et l'implémentation d'un lac de données ainsi que de son système de gestion des métadonnées, dans le but de stocker, organiser et exploiter des données hétérogènes**, et ce dans le contexte métier de l'habitat social. Cette problématique générale peut se scinder en des objectifs plus concrets.

Premièrement, nous avons pour but de proposer une modélisation des métadonnées pour leur organisation au sein du lac de données. C'est en effet une thématique de recherche en vogue, et nous souhaitons nous positionner avec un modèle très générique, qui permet de s'adapter à différents cas d'usages, en permettant notamment de gérer des données de tous types et de pouvoir les classer à travers des catégorisations métier diverses.

Avec un tel modèle de métadonnées, nous pouvons ensuite nous pencher sur la question de la conception d'un lac de données dans sa globalité. Nous devons pour cela définir d'une part son architecture fonctionnelle qui décline les composants nécessaires au bon fonctionnement du lac, et d'autre part implémenter le lac avec des technologies existantes.

Enfin, une fois le lac de données implémenté, notre objectif est de proposer des améliorations en ce qui concerne l'expérience utilisateur. Plus particulièrement, nous considérons que l'ingestion de nouvelles données dans le lac est une tâche primordiale, mais qui peut aussi être fastidieuse, et nous souhaitons assister l'utilisateur pour l'aider lors de cette étape, dans l'optique que l'ingestion devienne plus simple.

1.3 Contributions et organisation du manuscrit

Pour traiter la problématique et atteindre les trois objectifs, nous proposons un manuscrit composé de six chapitres. Le chapitre d'introduction générale constitue le premier

chapitre de ce manuscrit, et les cinq autres sont agencés de la manière suivante. Dans le **chapitre 2**, nous présentons l'état de l'art autour des lacs de données. Nous commençons par présenter le concept de lac de données au sens large, en donnant l'évolution de la définition du concept au fil des années, pour finalement introduire notre définition qui servira de référence pour tout le manuscrit. Nous discutons ensuite des métadonnées du lac de données, en définissant ce qu'elles doivent représenter, et comment elles permettent de décrire efficacement les données enregistrées dans le lac. Ceci nous permet de nous pencher plus en détail sur les systèmes de métadonnées proposés dans la littérature, en faisant un focus sur les modèles de métadonnées, qui visent à être plus génériques et reproductibles. Finalement, nous présentons les architectures et les technologies existantes pour mettre en œuvre de manière concrète un lac de données.

Le **chapitre 3** présente nos contributions concernant la modélisation des métadonnées d'un lac de données. Nous présentons notre premier modèle de métadonnées, baptisé MEDAL, qui se base sur une typologie de métadonnées en trois catégories (intra, inter et globales). Au niveau logique, MEDAL adopte une représentation à base de graphes. Ensuite, nous introduisons une évolution de MEDAL qui est notre métamodèle de métadonnées, nommé goldMEDAL. Il se distingue de son prédécesseur par un plus haut niveau d'abstraction, ce qui le rend à la fois plus générique et plus simple. Il se compose d'un métamodèle conceptuel et d'un métamodèle logique, ce dernier pouvant être instancié puis traduit en des modèles physiques.

Le **chapitre 4** donne notre proposition d'architecture de lac de données, basée sur quatre composants, et qui fonctionne de manière cyclique, insistant ainsi sur la nécessité d'utiliser le lac de données comme un système complet et « vivant », et non simplement un dépôt de données ou un espace d'analyse. Nous nous appuyons sur cette architecture pour proposer ensuite notre implémentation de lac de données, appelée HOUDAL, qui est dédiée à la problématique métier de l'habitat social. HOUDAL dispose d'une interface avec laquelle l'utilisateur peut interagir, et utilise un modèle physique de goldMEDAL pour gérer les métadonnées.

Le **chapitre 5** présente nos travaux portant sur la création semi-automatique de métadonnées pour les données structurées. Cette contribution découle d'un nouveau besoin ancré dans le contexte métier de l'habitat social, où les bailleurs génèrent de manière périodique des fichiers de données structurées, mais dont le schéma est susceptible d'évoluer lors de nouvelles occurrences de données. Nous avons créé un assistant intitulé QSTR qui vise à assister l'utilisateur dans la création de liens entre les différentes occurrences des mêmes données, et ce malgré un schéma pouvant différer. Plus particulièrement, QSTR effectue des comparaisons entre les attributs des nouvelles données avec ceux des données déjà enregistrées dans le lac, et effectue ensuite des suggestions de liens entre attributs à l'utilisateur, qu'il doit alors valider ou invalider. Les informations générées à l'aide de QSTR sont modélisées avec le métamodèle goldMEDAL et enregistrées dans le système de métadonnées du lac.

Enfin, dans le **chapitre 6**, nous dressons un bilan de nos travaux et de nos contributions. Nous finissons par ouvrir le champ des possibles en présentant des perspectives de recherche qui surviennent à la lumière de nos travaux.

Chapitre 2

État de l'art

Sommaire

"A delusion starts like any other idea, as an egg. Identical on the outside, perfectly formed. From the shell, you'd never know anything was wrong. It's what's inside that matters."

Narrator, *Legion (2018)*

Résumé

Dans ce chapitre, nous présentons un état de l'art sur les lacs de données et sur les métadonnées. Le concept de lac étant assez récent, plusieurs travaux ont été menés en parallèle, en particulier sur l'organisation des métadonnées au sein du lac, ainsi que sur la mise en œuvre concrète d'un lac de données. Toutefois, il n'existe pas encore de véritable consensus scientifique autour du concept, de l'organisation et de la nature des métadonnées, ou encore des architectures de lac de données, c'est pourquoi plusieurs points de vue existent dans la littérature.

Nous commençons par discuter du concept de lac de données ainsi que du positionnement d'autres concepts qui gravitent autour du terme de lac de données. Nous étudions ensuite quelles métadonnées sont pertinentes pour décrire au mieux les données, en distinguant des métadonnées universelles pour tous types de données, mais aussi des métadonnées spécifiques selon le type des données. Après cela, nous nous penchons sur la question de l'organisation des métadonnées d'un lac de données, en mettant l'accent sur les modèles de métadonnées. Enfin, nous passons en revue les différentes architectures et technologies existantes pour la mise en œuvre concrète d'un lac de données.

2.1 Introduction

Le concept de lac de données a été proposé en 2010 par James Dixon en réponse aux limites des systèmes d'entreposage et d'exploitation des données, à savoir les entrepôts de données [Dixon, 2010]. Rapidement, le lac de données a été considéré comme une approche novatrice, et de nombreuses équipes de recherche ont commencé à travailler sur des problématiques autour de ce concept [Miloslavskaya and Tolstoy, 2016, Madera and Laurent, 2016, Suriarachchi and Plale, 2016].

Un lac de données est défini par une approche *schema-on-read*, c'est-à-dire que les données sont enregistrées sans schéma prédéfini, et ce dernier n'est spécifié qu'à l'interrogation des données. Cette approche permet donc d'enregistrer des données de tous types, mais oblige aussi à disposer d'un ensemble de métadonnées efficace pour éviter que le lac de données ne se transforme en marécage de données, i.e. un lac de données inutilisable. En outre, les métadonnées sont aussi primordiales puisqu'elles assurent la gouvernance des données, et permettent donc de suivre le cycle de vie des données, assurer la sécurité des données et garantir leur qualité. Le premier problème de recherche qui nous intéresse ici porte donc sur les informations qui constituent les métadonnées du lac, pour décrire au mieux les données présentes dans le lac. En effet, bien que tous les ensembles de données puissent être décrits par des informations universelles, des données structurées ou non structurées, des images ou des documents textuels, sont amenées à posséder des métadonnées spécifiques pour décrire leur contenu.

Un second problème primordial concerne l'organisation des métadonnées du lac de données. En effet, puisqu'un lac nécessite des métadonnées pour fonctionner, l'étude de leur organisation au sein d'un système de métadonnées est un sujet de recherche d'actualité, et plusieurs propositions ont été formulées à cet égard. La littérature propose ainsi des systèmes de métadonnées qui sont soit des implémentations de lac de données adaptées à un cas d'usage spécifique, soit des modèles de métadonnées, plus explicites, mais qui doivent pouvoir être implémentés et s'adapter à des cas d'usages différents.

La mise en œuvre concrète d'un lac de données est un troisième problème de recherche largement abordé dans la littérature scientifique. De nouveaux outils issus de l'univers des mégadonnées, les bases de données NoSQL et Apache Hadoop [1] en tête, ont apporté de nouvelles approches pour implémenter des systèmes d'information environ en même temps que la première proposition du concept de lac de données par Dixon. C'est pourquoi plusieurs chercheurs se sont penchés sur comment implémenter le concept de lac de données à l'aide de technologies existantes, et ont donc travaillé sur les architectures de lacs de données, c'est-à-dire sur l'organisation des composants d'un lac, ainsi que les interactions qu'ils ont entre eux.

Le reste de ce chapitre est organisé comme suit. Nous commençons par présenter dans la section 2.2 les différentes définitions du concept de lac de données que nous pouvons trouver dans la littérature. Après une discussion sur ces définitions, nous introduisons notre propre définition. Nous présentons ensuite comment un lac se positionne par rapport à d'autres systèmes, notamment les entrepôts de données. Dans un second temps, la

1. `https://hadoop.apache.org/`

section 2.3 présente les métadonnées pertinentes à constituer pour décrire les données ingérées par le lac, en distinguant celles qui sont universelles pour toutes les données, et celles qui sont spécifiques selon le type des données. Par la suite, la section 2.4 explicite les enjeux de la gestion des métadonnées dans un lac de données, et compare plusieurs systèmes de métadonnées de la littérature selon les fonctionnalités qu'ils proposent. Nous nous penchons ensuite plus en détail sur les modèles de métadonnées, qui sont explicites et réutilisables. Nous exposons dans la section 2.5 les différentes architectures existantes pour la mise en place d'un lac de données, ainsi que les technologies qui permettent d'implémenter un lac. Finalement, la section 2.6 conclut cet état de l'art.

2.2 Le concept de lac de données

Avant d'étudier en détail les lacs de données et les travaux récents sur le sujet, il est primordial de définir clairement ce qui se cache derrière cette appellation. C'est en effet un terme assez nouveau, surtout en comparaison à d'autres concepts en informatique qui existent depuis plusieurs décennies, comme les entrepôts de données. Il arrive encore que le concept de lac de données soit utilisé à tort, notamment par des acteurs industriels à des fins de marketing, mais aussi par un simple manque de compréhension face à une multitude de définitions contradictoires. Pour débuter cet état de l'art, nous tâchons donc d'éclaircir les ambiguïtés qui gravitent autour de la notion de lac de données.

Nous commençons par présenter dans la section 2.2.1 plusieurs définitions du concept de lac de données issues de la littérature. Après les avoir discutées, nous donnons notre définition de ce qu'est un lac de données. La section 2.2.2 donne les principales différences entre un entrepôt de données et un lac de données, ainsi que comment lacs et entrepôts peuvent coexister. Enfin, nous discutons dans la section 2.2.3 des autres approches qui gravitent autour du concept de lac de données, en particulier des termes très récents, qui parfois utilisent le terme de lac de données de manière détournée.

2.2.1 Définitions

2.2.1.1 Dans la littérature

Le concept de lac de données a été introduit par [Dixon, 2010] comme une alternative aux magasins de données (*data marts*), qui sont des sous-ensembles des entrepôts, auxquels il reproche de mettre les données en silos. Il le définit à travers la métaphore suivante : « Si vous considérez un *data mart* comme un magasin d'eau en bouteilles - nettoyée, emballée et structurée pour une consommation facile - le lac de données est un grand plan d'eau dans un état plus naturel. Le contenu du lac de données afflue d'une source pour remplir le lac, et divers utilisateurs du lac peuvent venir l'examiner, y plonger ou prendre des échantillons ». Ainsi, le lac de données tel que conçu par Dixon est un vaste dépôt de données brutes de structures hétérogènes, alimenté par des sources de données externes et à partir duquel des analyses diverses peuvent être réalisées.

Suite à la définition de Dixon, certains travaux ont rapidement associé le lac de données à la technologie Hadoop [O'Leary, 2014, Fang, 2015]. Ainsi, Fang considère le lac

de données comme une méthodologie consistant à utiliser des technologies libres ou peu coûteuses, typiquement Hadoop, pour assurer le stockage, le traitement et l'exploration des données brutes au sein d'une entreprise. Cependant, cette vision est de plus en plus minoritaire dans la littérature, le concept de lac de données étant désormais également associé à des solutions propriétaires comme Azure ou IBM [Madera and Laurent, 2016, Sirosh, 2016].

Une définition plus consensuelle consiste à voir un lac de données comme un dépôt central où des données de tous formats sont stockées sans schéma strict [Laskowski, 2016, Khine and Wang, 2017, Mathis, 2017]. Cette définition est basée sur deux caractéristiques clés du lac de données : la variété des données et l'approche *schema-on-read*. La variété des données désigne le fait d'intégrer des données de tous types, et donc hétérogènes. La propriété *schema-on-read* consiste à définir le schéma des données seulement lors de leur analyse [Miloslavskaya and Tolstoy, 2016]. Nous parlons aussi d'approche *late binding*, à l'inverse de l'approche *early binding* des systèmes d'information décisionnels traditionnels [Fang, 2015].

Toutefois, il convient de noter que la définition basée sur la propriété *schema-on-read* et la variété des données, bien que consensuelle, demeure incomplète dans le sens où elle donne peu de détails sur les caractéristiques d'un lac de données. C'est pourquoi [Madera and Laurent, 2016] introduisent une nouvelle définition plus complète. Un lac de données est alors considéré comme une vue logique de toutes les sources de données et de tous les ensembles de données dans leur format brut, accessible par des *data scientists* ou statisticiens pour l'extraction de connaissances. Cette définition est complétée par une liste de caractéristiques clés :

1. la qualité des données dans le lac est assurée par un ensemble de métadonnées ;
2. le lac est contrôlé par une politique et des outils de gouvernance des données ;
3. l'utilisation du lac est limitée aux statisticiens et aux *data scientists* ;
4. le lac intègre des données de tous types et formats ;
5. le lac de données possède une organisation logique et physique.

2.2.1.2 Notre définition

La définition la plus complète des lacs de données est celle de [Madera and Laurent, 2016], qui en plus de la variété des données et de l'approche *schema-on-read*, définit des caractéristiques supplémentaires. Cependant, certaines caractéristiques proposées dans cette définition sont discutables de notre point de vue. En effet, les autrices réservent l'utilisation du lac aux spécialistes des données et excluent *de facto* les experts métiers pour des raisons de sécurité. Pourtant, il est tout à fait envisageable, selon nous, de donner un accès contrôlé à ce type d'utilisateurs à travers une plateforme de navigation ou d'analyse.

De plus, nous ne partageons pas la vision du lac de données comme une vue logique des sources de données, dans le sens où les sources de données sont parfois extérieures à l'entreprise, et donc au lac de données. C'est d'ailleurs ce qui ressort de la définition

initiale de [Dixon, 2010], qui précise que les données du lac proviennent de sources de données. L'inclusion des sources de données dans le lac, comme proposé par [Madera and Laurent, 2016], peut donc être considérée comme contraire à l'esprit des lacs de données.

Enfin, bien que plus complète, la définition de [Madera and Laurent, 2016] omet une propriété essentielle des lacs de données, que nous retrouvons notamment dans les définitions de [Miloslavskaya and Tolstoy, 2016] et [Haste, 2017] : la capacité de passage à l'échelle. En effet, un lac de données étant destiné au stockage et au traitement des mégadonnées, il est indispensable de traiter la problématique liée au volume des données en assurant le passage à l'échelle, sans quoi le lac de données serait peu pertinent.

Au vu de ce qui précède, nous proposons une nouvelle définition des lacs de données qui vise à être aussi complète que celle de [Madera and Laurent, 2016], mais plus conforme à notre vision du concept de lac de données. Notre définition amende celle de [Madera and Laurent, 2016] et introduit la caractéristique de passage à l'échelle. Dans tout le reste de ce document, nous considérons qu'un lac de données correspond à cette définition.

Définition 1 *Un lac de données est un système évolutif (en termes de passage à l'échelle) de stockage et d'analyse de données de tous types, dans leur format natif, utilisé* principalement *par des spécialistes des données (statisticiens, data scientists, data analysts) pour l'extraction de connaissances. Les caractéristiques d'un lac de données incluent :*

1. *un catalogue de métadonnées qui assure la qualité des données ;*
2. *une politique et des outils de gouvernance des données ;*
3. *l'ouverture à tous types d'utilisateurs ;*
4. *l'intégration de données de tous types ;*
5. *une organisation logique et physique ;*
6. *le passage à l'échelle.*

2.2.2 Lac de données et entrepôt de données

Lors de la première mention du terme de lac de données, Dixon suggérait que ce concept venait en alternative aux entrepôts de données [Dixon, 2010]. Maintenant que nous avons donné notre définition du concept de lac, nous pouvons désormais confronter les deux concepts de manière plus précise. Nous commençons par donner les principales différences (section 2.2.2.1), puis nous présentons comment ces deux éléments peuvent coexister (section 2.2.2.2).

2.2.2.1 Différences principales

L'entrepôt de données (*data warehouse*) est la pièce maîtresse d'un processus d'informatique décisionnelle [Inmon, 1996]. Son objectif ultime est de fournir aux décideurs une vue d'ensemble sur les données ainsi qu'une aide à la décision, dans le but final d'améliorer l'activité économique gérée par les décideurs [Watson and Wixom, 2007].

La première étape est de collecter les données, pouvant provenir de différentes sources, telles que des bases de données, des fichiers Excel, des fichiers CSV, des données d'ERP,

etc. À l'aide d'un processus d'ETL (*Extract, Transform, Load*), ces données vont être extraites de leur source d'origine, transformées puis chargées dans l'entrepôt de données. Cet entrepôt est dans la majorité des cas une base de données relationnelle, dont les tables sont organisées « en étoile » (parfois en flocon ou en constellation), avec une table de faits au centre, et une multitude de tables de dimensions autour, qui représentent les axes d'analyse sur lesquels l'utilisateur pourra explorer les faits.

Une fois l'entrepôt constitué et chargé, il faut analyser les données. L'analyse est réalisée d'une part grâce à l'exploration de rapports, créés par des outils de *reporting*, mais aussi grâce au traitement analytique en ligne, en anglais *OnLine Analytical Processing* (OLAP). Ensemble, ces deux techniques permettent d'explorer les données de l'entrepôt. Les rapports sont constitués de tableaux, graphiques et autres indicateurs et ont pour objectif de faire ressortir au mieux les informations présentes dans les données. Il existe souvent un rapport par secteur d'activité (ventes, ressources humaines, impayés, etc.).

Notons que la littérature scientifique propose de nouvelles approches pour constituer des entrepôts de données, en prenant notamment appui sur des bases de données NoSQL [Bicevska and Oditis, 2017], comme des bases de données orientées colonnes [Boussahoua et al., 2017] ou orientées documents [Bouaziz et al., 2019]. Toutefois, ces approches sont très peu utilisées en entreprise, et de ce fait, compte-tenu du contexte industriel de cette thèse, nous préférons exclure ces méthodes et ne considérer que les entrepôts de données « classiques » contenant des données structurées homogènes pour les comparer aux lacs de données.

Grâce à un rapide rappel sur les entrepôts de données, nous pouvons dresser une comparaison entre entrepôt et lac de données. Le premier élément marquant est qu'un entrepôt stocke généralement des données structurées, ou a minima de nature homogène, tandis qu'un lac de données peut stocker des données de tous types. Ainsi, le spectre des données qu'il est possible de traiter par ces deux systèmes n'est pas le même : il est plus large pour le lac de données. Il est aussi intéressant de noter que le stockage des données est souvent - mais pas systématiquement - moins coûteux dans un lac de données, mais ceci dépend entièrement des technologies utilisées.

La seconde différence principale concerne l'approche propre à chaque système pour l'enregistrement des données. Dans l'entrepôt de données, le schéma est défini à l'enregistrement, c'est une approche *schema-on-write* (ou *early binding*). Ceci signifie que les données sont transformées avant d'être enregistrées, d'où l'utilisation d'un processus ETL. À l'inverse, un lac de données fonctionne en mode *schema-on-read* (ou *late binding*), c'est-à-dire que les données ne sont transformées qu'à l'interrogation et non à l'enregistrement : c'est un fonctionnement en ELT (*Extract – Load - Transform*). La conséquence de ce fonctionnement est qu'il n'y a pas de perte d'information lors de l'enregistrement des données dans le lac.

Enfin, la troisième différence entre les deux approches découle des deux précédentes : un lac de données est bien plus flexible qu'un entrepôt de données, qui est très rigide. En effet, pouvoir stocker des données de tous types dans leur format natif demande plus de souplesse qu'un entrepôt qui ingère des données dont la modélisation est définie à

l'avance. Toutefois, la flexibilité du lac de données est à double tranchant, puisqu'un lac a obligatoirement besoin d'un système de métadonnées fonctionnel et efficace, sous peine de se transformer en un marécage de données inexploitable [Miloslavskaya and Tolstoy, 2016].

Enfin, nous pouvons donner une quatrième différence entre entrepôt de données et lac de données, qui porte sur les analyses qu'il est possible de mener. Dans les entrepôts, les données sont transformées avant enregistrement pour répondre à un schéma prédéfini. La modélisation des données est guidée par des besoins d'analyse prédéfinis. À l'inverse, dans les lacs de données, les données sont enregistrées dans leur format brut, puisque les besoins d'analyse ne sont pas définis à l'avance. Ainsi, le lac de données laisse la porte ouverte à tous types d'analyses.

Pour dresser un récapitulatif, nous listons ces différences entre entrepôt et lac dans le tableau 2.1.

Critère \ Système	**Entrepôt de données**	**Lac de données**
Type de données stockées	Données structurées (ou de nature homogène)	Données de tous types
Intégration des données	ETL (*Extract - Transform - Load*)	ELT (*Extract - Load - Transform*)
Définition du schéma	À l'insertion (*schema-on-write*)	À l'interrogation (*schema-on-read*)
Flexibilité	Rigide, peu évolutif	Flexible, mais besoin d'un système de métadonnées
Type d'analyses	Analyses prédéfinies	Tous types d'analyses

Table 2.1 – Différences principales entre entrepôt de données et lac de données

2.2.2.2 Cohabitation

Entrepôt de données et lac de données sont des systèmes bien différents, mais qui peuvent cohabiter au sein d'un même périmètre. Une première possibilité est de considérer qu'un lac de données sert de source à un entrepôt de données. Ce cas de figure a pu être rencontré à de nombreuses reprises, notamment lors de l'émergence du concept de lac de données, mais plutôt en considérant le lac uniquement comme un espace de stockage de données volumineuses [Fauduet and Peyrard, 2010]. C'est notamment une vision reprise pour le nouveau concept de *Lakehouse* (que nous présentons dans la section 2.2.3.1). La figure 2.1 représente cette configuration de manière schématique, où un lac de données sert de source à un entrepôt de données (le triangle représente d'autres potentiels systèmes en aval du lac).

À l'inverse, nous pouvons aussi imaginer un entrepôt de données servant de source au lac de données. Dans le cadre de la mise en place d'un nouveau lac de données comme

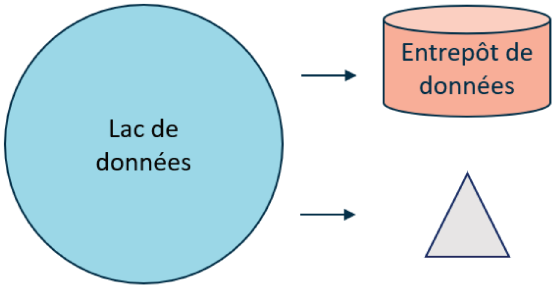

FIGURE 2.1 – Un lac de données comme source d'un entrepôt de données

source unique de données pour toutes les applications, il est tout à fait envisageable qu'un entrepôt de données, existant depuis plus longtemps, puisse alimenter le lac en données. L'idée serait ainsi de migrer les données de l'entrepôt dans le lac et les exploiter en les combinant à d'autres données. La figure 2.2 décrit ce mode de fonctionnement, où un entrepôt de données alimente un lac de données (le triangle représente d'autres potentielles sources en amont du lac).

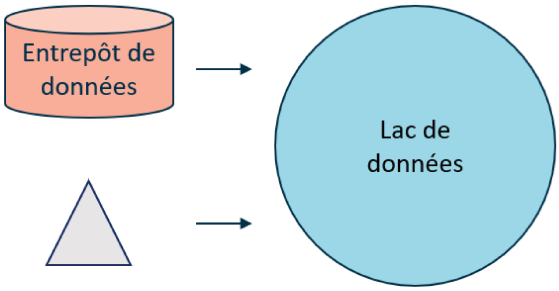

FIGURE 2.2 – Un entrepôt de données comme source d'un lac de données

Finalement, une troisième possibilité est qu'un entrepôt de données soit situé directement dans le lac [Inmon, 2016]. En effet, puisqu'un lac de données peut contenir des données de tous types, il est tout à fait possible d'y insérer les données d'un entrepôt, qui sont souvent des tables provenant d'une base de données relationnelle. Il est même possible de décrire à travers les métadonnées que les tables en question sont au sein d'un modèle en étoile, si nécessaire. Contrairement aux précédentes approches, il n'y a ici qu'un seul système, le lac de données ; l'entrepôt de données n'est pas un système à part entière, mais il est une partie du lac. C'est cette approche qui respecte au mieux, selon nous, le concept de lac de données tel que nous l'avons défini. La figure 2.3 illustre ce mode de fonctionnement.

2.2.3 Approches connexes

Comme la notion de lac de données est encore très récente, des éditeurs s'en sont emparés pour mettre en valeur des solutions de leur création. Toutefois, ces nouvelles

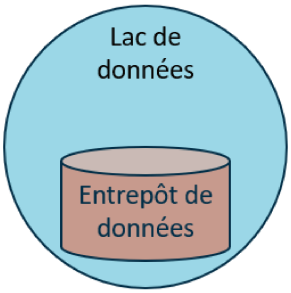

FIGURE 2.3 – Un entrepôt de données à l'intérieur d'un lac de données

solutions font parfois référence à des lacs de données dont la définition varie considérablement de la nôtre. Pour éclaircir cela, nous détaillons ci-après ce qui se cache derrière les différents termes qui gravitent autour de celui de lac de données.

2.2.3.1 Lakehouse

Un terme ayant émergé récemment est celui de *lakehouse*, formé à partir de *data warehouse* et *data lake* [Armbrust et al., 2021]. Les auteurs désignent le *lakehouse* comme une nouvelle approche pour le stockage, l'organisation et l'exploitation des données, en la comparant aux approches existantes. Ils rappellent que les premières approches étaient basées principalement sur un entrepôt de données, alimenté par des données structurées qui sont transformées par un processus ETL (*schema-on-write*) avant d'être insérées dans l'entrepôt. Toutefois, ces systèmes ne peuvent contenir que des données structurées ou homogènes, et ont donc été bouleversés par l'émergence des mégadonnées et de leur variété.

Les auteurs précisent ensuite que les lacs de données (au sens d'un système de stockage de données massives, et non au sens de notre définition) ont fait leur apparition, et permettent d'ingérer différents types de données sans schéma prédéfini (*schema-on-read*). Dans beaucoup de cas, un lac est donc placé en amont d'un entrepôt de données, et ce dernier est alimenté en données structurées depuis le lac via un processus ETL. Les auteurs relèvent toutefois qu'avoir à gérer un lac et un entrepôt séparément s'avère complexe à exploiter et à maintenir. Le *lakehouse* veut ainsi éviter cette dualité en ne proposant qu'un lac de données, et qui sert de base pour toutes les analyses. Dans cette optique, il est proposé qu'un entrepôt de données soit situé au sein du lac de données, et cette inclusion transforme le lac en *lakehouse*. Ainsi, il sert de source à tous types d'analyses sur les données qu'il contient.

La figure 2.4 illustre cette évolution des plateformes d'analyse de données. À gauche, les auteurs décrivent le fonctionnement classique d'un processus ETL qui alimente un entrepôt de données à partir de données sources, pour fournir des analyses décisionnelles et des rapports. Au centre, diverses sources de données alimentent un lac de données, lui-même alimentant un entrepôt de données à travers un processus ETL. Le lac de données peut être exploité par des analyses de science de données, tandis que c'est l'entrepôt

de données qui est interrogé pour des analyses d'informatique décisionnelle. Il est intéressant de noter que cette approche est celle que nous avons décrite dans la figure 2.1. Enfin, à droite de la figure est illustré le *lakehouse*, où un lac de données ingère tous types de données pour les mettre à disposition de diverses applications. Bien que ce ne soit pas visible sur la figure, les auteurs précisent que le lac de données peut contenir un entrepôt de données, d'où la présence d'un processus ETL au sein du lac. Cette approche correspond à celle que nous avons décrite dans la figure 2.3.

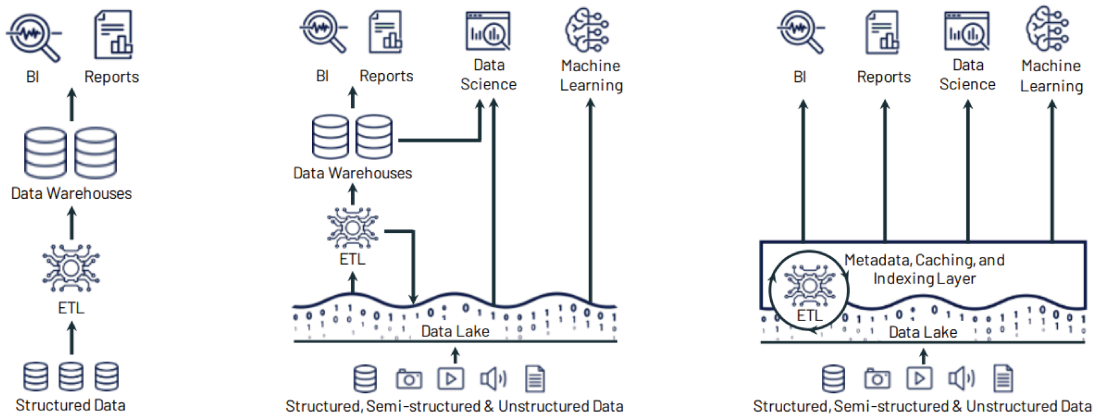

FIGURE 2.4 – Evolution des plateformes d'analyse de données (image tirée de [Armbrust et al., 2021])

Il est intéressant de noter que cette vision du *lakehouse* colle parfaitement avec notre définition du concept de lac de données, introduite en section 2.2.1.2. Nous voyons en effet le lac de données comme l'élément central de toute architecture pour l'organisation et le stockage de données hétérogènes. La différence de terme peut venir du fait de la confusion, encore très présente, entre le concept de lac de données tel que nous l'avons défini et le « *data lake* » réduit à un espace de stockage distribué et peu coûteux pour des fichiers de données. Une autre piste suggère que l'introduction du terme *lakehouse* a un objectif marketing, pour justement se dégager de la confusion autour du lac de données et viser à proposer un concept nouveau.

2.2.3.2 Data Mesh

Le maillage de données (*Data Mesh*) est un autre terme ayant fait une apparition récente, et qui gravite autour de la notion de lac de données [Dehghani, 2019]. Ce concept est une réponse à la dichotomie entre les données opérationnelles et les données analytiques. Les premières se trouvent dans des bases de données, sont interrogées en général de manière transactionnelle par des microservices et servent à alimenter les applications métier. Les données analytiques, quant à elles, sont une vision temporelle et agrégée des faits de l'entreprise, souvent modélisées pour fournir des informations rétrospectives ou prévisionnelles. Elles servent souvent à entraîner des algorithmes d'apprentissage automatique

ou alimentent des rapports analytiques, et ont pour objectif d'optimiser l'activité. Dans de nombreux cas, les données analytiques passent à travers des lacs de données et/ou des entrepôts de données, et visent ensuite à fournir les données à divers utilisateurs. Par exemple, des scientifiques des données sont susceptibles d'interroger le contenu du lac, alors que des contrôleurs de gestions peuvent exploiter les données de l'entrepôt [Dehghani, 2020]. Ce mode de fonctionnement a tendance à créer un grand nombre de transformations de données, ce qui mène rapidement à un système difficilement maintenable et exploitable.

L'idée derrière le maillage de données est donc de pallier ce problème en évitant de dissocier données opérationnelles et données analytiques. Pour cela, les données sont séparées dans des nœuds de maillage (*Mesh nodes*), et donc potentiellement enregistrées dans des espaces physiques différents, mais une gouvernance des données centralisée permet de s'y retrouver facilement [Machado et al., 2021]. Le maillage de données est basé sur quatre piliers principaux.

- Le premier pilier du maillage de données consiste à découper les informations en **domaines**. Dans une entreprise, nous retrouvons généralement plusieurs équipes, qui n'ont pas les mêmes besoins en termes d'exploitation des données. Cependant, différentes équipes peuvent avoir besoin d'utiliser les mêmes données sources, mais les retravailler différemment, par exemple à des niveaux d'agrégation différents. Puisque les équipes disposent de leurs domaines respectifs, elles peuvent prendre les données dont elles ont besoin et les retravailler à leur guise au sein de leur domaine, pour répondre à leurs besoins.
- Considérer les **données en tant que produit** est le second axe qui constitue le maillage de données. Ceci prend appui sur les principes du *product thinking*, qui stipulent qu'un bon produit doit apporter de la valeur (répondre au besoin métier du domaine), être utilisable (i.e., que son utilisation est évidente) et réalisable (avec les outils et technologies disponibles). Cette approche garantit aussi la bonne prise en compte du cycle de vie des données, et que les données du domaine sont de bonne qualité.
- Le troisième principe d'un maillage est de proposer une **plate-forme de données en libre-service** en tant que support des « produits de données ». Chaque domaine doit pouvoir proposer l'accès aux données retravaillées qui servent à répondre au(x) besoin(s), à l'aide d'un micro-service. L'idée est que chaque domaine puisse à la fois exploiter les données sources, qui sont brutes, mais aussi les données prêtes à l'emploi mises à disposition par les autres domaines. En effet, il est tout à fait envisageable que des domaines différents partagent le besoin de traiter des données de la même manière. Dans ce cas, il n'est pas nécessaire d'effectuer le traitement dans tous les domaines, mais seul l'un d'entre eux effectue le traitement et met à disposition des autres les données traitées.
- Enfin, avoir une **gouvernance des données fédérée** est le dernier pilier du maillage de données. Puisque les différents domaines du maillage exploitent les données sources, et que les domaines sont aussi amenés à interagir entre eux, il est

primordial de suivre qui utilise quoi et comment. La gouvernance des données fédérée doit permettre aux utilisateurs de connaître les données brutes à disposition, ainsi que les micro-services proposés par les domaines. Ainsi, bien que le maillage de données soit basé sur une décentralisation des données et de leurs usages, la gouvernance des données doit elle être centralisée, pour avoir une vue d'ensemble sur les interactions entre les données et les domaines.

La figure 2.5 donne un exemple de maillage de données. Les données sources sont représentées au bas de la figure (en bleu), et le maillage contient deux domaines. Le domaine 2 (en vert) exploite à la fois des données brutes et des données mises à disposition par le domaine 1 (en violet). Aussi, la présence d'une gouvernance des données fédérée au sein du maillage est explicitée (en jaune).

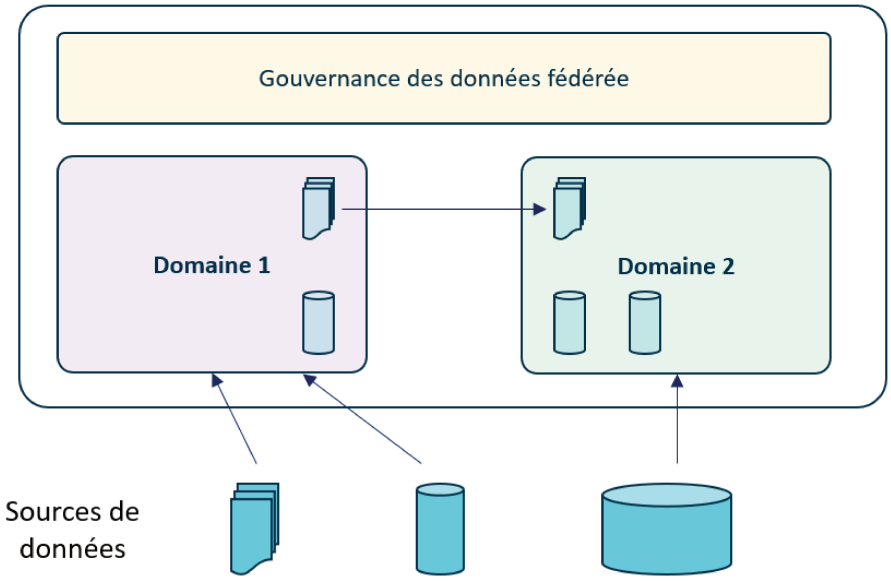

FIGURE 2.5 – Exemple de maillage de données (*Data Mesh*)

Nous considérons que le maillage de données tel que présenté ici est proche de notre vision d'un lac de données. Le fait de regrouper les données au sein d'un même domaine fait référence à une architecture de lac de données basée sur la sémantique des données, que nous présentons dans la section 2.5. En considérant un ensemble de données comme un produit, c'est-à-dire en prenant en considération tout son cycle de vie, cela revient à capter des métadonnées, en particulier le lignage des données. Enfin, la mise à disposition des données en libre-service, et une centralisation de la gouvernance des données, sont justement deux points cruciaux de notre définition d'un lac de données. En somme, un maillage de données se trouve finalement assez proche de notre définition de lac de données. L'émergence de ce nouveau terme peut à nouveau provenir d'une mauvaise compréhension du concept de lac de données et/ou d'un choix marketing pour introduire un nouveau terme tendance.

2.3 Description des données à travers les métadonnées

Le concept de lac de données prône un nouveau paradigme de stockage et d'analyse des données à travers une approche *schema-on-read*, c'est-à-dire une définition *a posteriori* du schéma des données. Sans schéma prédéfini, un système de métadonnées efficace est alors un élément indispensable pour rendre les données interrogeables, et empêcher ainsi le lac de se transformer en *data swamp* [Miloslavskaya and Tolstoy, 2016]. Un problème de recherche important concerne donc le contenu des métadonnées d'un lac de données, c'est-à-dire les informations qu'elles portent pour décrire au mieux les données du lac. Dans cette section, nous étudions quelles sont les informations pertinentes pour constituer les métadonnées lors de l'ingestion des données dans le lac.

Nous commençons par discuter dans la section 2.3.1 de la définition du terme métadonnée au sens large, puis plus précisément en informatique. Nous étudions aussi quelles métadonnées existent dans différents domaines de l'informatique. Ensuite, nous nous focalisons sur les lacs de données et présentons dans la section 2.3.2 les métadonnées qui sont plutôt universelles, et permettent de décrire tous types de données. La constitution de ces métadonnées permet d'avoir une uniformité pour décrire toutes les données de la même manière. Enfin, dans la section 2.3.3, nous abordons la question des métadonnées spécifiques à chaque type de données, dans l'optique de décrire au mieux les données ingérées dans le lac.

2.3.1 Les métadonnées en informatique

Le terme de *métadonnée* peut porter plusieurs significations. Étymologiquement, le préfixe méta désignant l'auto-référence indique que les métadonnées sont « des données sur les données », c'est-à-dire des données qui décrivent d'autres données. De manière un peu plus précise en informatique, les métadonnées possèdent deux caractéristiques principales. D'une part, elles viennent en complément des données qu'elles décrivent, et donc ne peuvent pas exister sans elles. D'autre part, elles sont relativement structurées, tandis que les données qu'elles décrivent peuvent ne pas l'être. [Riley, 2017] s'appuie sur ces éléments pour proposer sa définition des métadonnées, que nous partageons : « les métadonnées sont des informations structurées qui décrivent, expliquent, localisent ou encore facilitent ou permettent la recherche, l'utilisation ou la gestion d'une ressource d'information ».

Dans le monde de l'informatique, les métadonnées sont partout. À titre d'exemple, dans l'explorateur de fichiers Windows utilisé par des milliards de personnes, l'intitulé d'un fichier, ou encore sa taille, constituent des métadonnées. Elles permettent de rendre les données plus compréhensibles en ajoutant une couche de contexte qui leur donne du sens. De même, l'identifiant de l'utilisateur ayant inséré les données et la date de création sont des métadonnées qui permettent de savoir les actions effectuées par qui et quand sur les fichiers concernés.

Les métadonnées sont aussi présentes dans les entrepôts de données [Wrembel and Bębel, 2007]. Par exemple, les processus ETL génèrent pour la grande majorité des *logs* décrivant le déroulement des différentes étapes d'extraction, de transformation ou de

chargement des données. De même, les documents du web sémantique possèdent des métadonnées susceptibles d'être extraites et exploitées [Ding et al., 2004].

Compte-tenu de la nature même d'un lac de données, à savoir l'entreposage de données de tous types sans schéma prédéfini, les métadonnées ont une importance capitale pour le bon fonctionnement du lac. À ce titre, les métadonnées doivent répondre à plusieurs besoins, dont nous dressons une liste ci-dessous [Halevy et al., 2016, Mehmood et al., 2019, Subramaniam et al., 2021].

- L'**accès aux données** constitue un premier besoin vital au sein du lac puisque les utilisateurs doivent pouvoir retrouver les données dont ils ont besoin.
- Il doit cependant y avoir une **différenciation de l'accès aux données** selon les utilisateurs, car certaines données peuvent être sensibles et donc uniquement consultables par les utilisateurs autorisés.
- Les métadonnées doivent assurer la **compréhensibilité des données**, c'est-à-dire permettre à l'utilisateur de comprendre le sens et le contexte des données qu'il manipule.
- De même, l'**auditabilité** désigne la nécessité de pouvoir déceler et comprendre d'éventuelles incohérences dans les données du lac, en retraçant l'historique des traitements effectués sur celles-ci.
- Ceci s'accompagne du besoin de **reproductibilité des analyses**, qui exprime que tout traitement effectué sur les données du lac doit pouvoir être reproduit à l'identique.
- Enfin, l'**intégration des données** est le dernier besoin que les métadonnées doivent remplir, et désigne le fait de pouvoir connecter les données entre elles, en les reliant ou les regroupant.

Les métadonnées peuvent être créées de différentes manières. Une première approche est la saisie manuelle des informations par un utilisateur du lac. Cette approche s'avère lente et fastidieuse, mais est toutefois nécessaire pour certaines métadonnées qu'il serait très difficile de construire autrement, comme une description ou des catégorisations métiers. La seconde approche consiste à extraire de manière automatique des informations à l'aide d'algorithmes passant en revue les données. Selon leur nature, ces métadonnées extraites automatiquement peuvent ensuite nécessiter une intervention humaine pour les valider ou non. Le cas échéant, nous pouvons considérer que les métadonnées sont créées de manière semi-automatique, i.e. que l'utilisateur a juste à les vérifier et confirmer ou infirmer leur exactitude.

Des auteurs ont proposé diverses méthodes d'extraction automatique de métadonnées. Concernant les données structurées, Bogatu et al. ainsi que Fernandez et al. utilisent des méthodes statistiques pour comparer le contenu d'attributs différents, et chercher à détecter ceux qui présentent des similitudes [Bogatu et al., 2020, Fernandez et al., 2018]. Klettke et al. introduisent une démarche pour tracer les évolutions du schéma d'une collection de données semi-structurées au fil du temps [Klettke et al., 2017]. Diamantini et al. vont plus loin en proposant une approche en deux étapes qui peut s'appliquer à des

données hétérogènes [Diamantini et al., 2018]. La première étape consiste à extraire des mots-clés des données avec des méthodes adéquates (par exemple, les intitulés d'attributs pour des données structurées, ou bien des entités nommées pour des documents textuels). Ensuite, ces mots-clés sont comparés à l'aide d'une ontologie, ou bien avec la distance de N-gram [Kondrak, 2005], puis reliés entre eux si une forte similarité est détectée, autant syntaxiquement que sémantiquement.

2.3.2 Métadonnées universelles pour tous types de données

Puisqu'un lac de données peut enregistrer des données de tous types, cela oblige à trouver des métadonnées universelles, ou transverses, qui peuvent s'adapter à tous les formats [Madera and Laurent, 2016, Miloslavskaya and Tolstoy, 2016]. L'objectif de ces métadonnées est de décrire de manière uniforme toute les données présentes dans le lac. En proposant des informations universelles, l'utilisateur peut s'y retrouver plus facilement.

Oram introduit une première classification de métadonnées avec trois catégories [Oram, 2015]. Les métadonnées sont rassemblées selon leur rôle, mais aussi selon la manière dont elles sont générées.

- Le premier niveau est celui des **métadonnées business** (*business metadata*), qui sont les plus abstraites et décrivent les données pour les rendre plus compréhensibles. Il s'agit en général du nom des attributs ainsi que des contraintes d'intégrité, dans le cas de données structurées. En général, elles sont définies par l'utilisateur lors de l'ingestion des données au sein du lac.
- La seconde catégorie est celle des **métadonnées opérationnelles** (*operational metadata*), qui sont généralement créées de manière automatique lors du traitement des données. Ces métadonnées contiennent notamment des informations sur les données source et cible, la taille des données, ou encore le nombre d'enregistrements affectés par le traitement.
- Finalement, le troisième niveau est celui des **métadonnées techniques** (*technical metadata*), qui sont les plus fines et expriment comment les données sont représentées. Elles donnent notamment le format, la structure ou le schéma des données, si applicable. Là encore, ces métadonnées sont généralement créées automatiquement, soit par extraction à l'aide d'outils dédiés, soit directement depuis les informations fournies par le système source.

Par la suite, Diamantini et al. se sont basés sur cette typologie de métadonnées pour aller plus loin [Diamantini et al., 2018]. Ils montrent que la frontière entre ces trois catégories de métadonnées peut parfois être poreuse, et que certaines informations peuvent se retrouver dans deux catégories. Par exemple, le format des données peut être considéré à la fois comme des métadonnées opérationnelles et techniques.

La figure 2.6 illustre cette typologie de métadonnées, amendée par Diamantini et al., en incluant les intersections entre les catégories. Les trois catégories de métadonnées, respectivement business, opérationnelles et techniques sont représentées par des cercles de couleur bleue, verte et orange, respectivement.

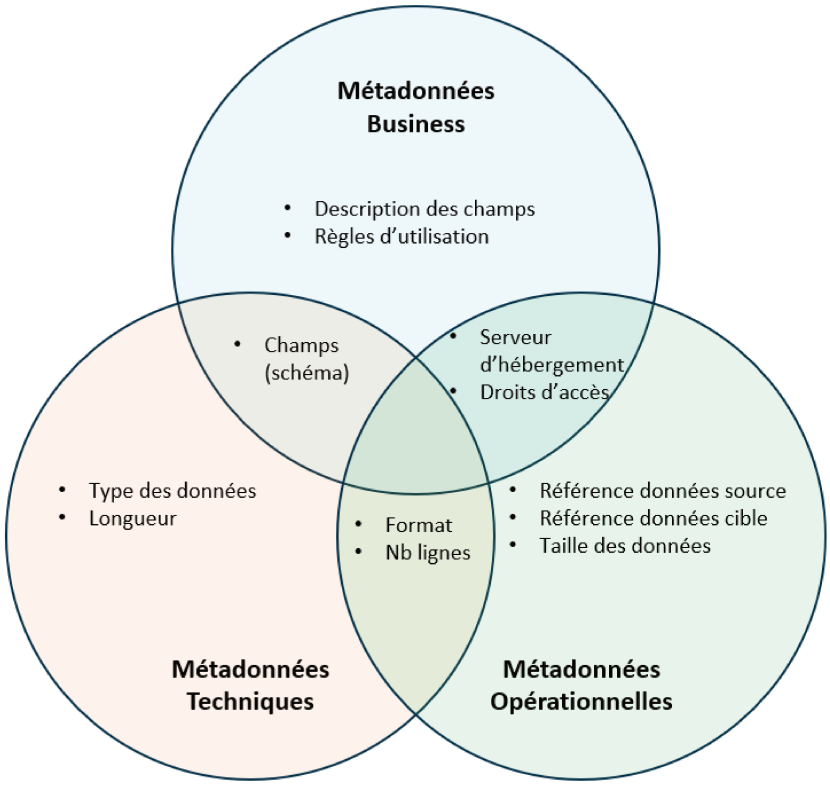

FIGURE 2.6 – Typologie de métadonnées business - techniques - opérationnelles de [Diamantini et al., 2018]

Ravat et Zhao proposent de classifier les métadonnées en deux catégories : les métadonnées intra et inter [Ravat and Zhao, 2019b]. Les métadonnées intra décrivent les jeux de données (*dataset*) présents dans le lac. Elles permettent aux utilisateurs de mieux comprendre les données avec lesquelles ils interagissent. Plusieurs types d'informations composent les métadonnées intra :

- Les **caractéristiques de données** comprennent diverses informations : un identifiant, le nom, la taille, la structure et la date de création des ensembles de données. Ces informations, assez simples, permettent aux utilisateurs de se faire une idée générale d'un ensemble de données.
- Les **métadonnées définitionnelles**, ou métadonnées de définition, précisent la signification des jeux de données. Deux sous-catégories sont définies : les métadonnées sémantiques et schématiques. Pour illustrer, les données peuvent être décrites sémantiquement par un texte ou par des mots-clés. D'un point de vue schématique, un ensemble de données structuré peut être présenté par un schéma de base de données.
- Les **métadonnées de navigation** contiennent l'emplacement des ensembles de données. Ce sont, par exemple, les chemins d'accès aux fichiers et/ou les URL de

connexion aux bases de données.

- Le **lignage** retrace le cycle de vie des données en contenant la source originale des jeux de données ainsi que l'historique des traitements effectués. Il permet de rendre les jeux de données plus fiables.
- Les **métadonnées d'accès** enregistrent les interactions entre les utilisateurs et les jeux de données. Il est alors possible de déterminer un niveau de confiance dans un ensemble de données si ce dernier a été beaucoup consulté par plusieurs utilisateurs.
- Les **métadonnées de qualité** visent à garantir la fiabilité d'un ensemble de données, en contrôlant sa cohérence et son exhaustivité.
- Les **métadonnées de sécurité** concernent la sensibilité des données et le niveau d'accès. En effet, un lac de données stocke des ensembles de données provenant de diverses sources, et certains peuvent contenir des informations sensibles auxquelles seuls des utilisateurs spécifiques peuvent accéder. Les métadonnées de sécurité peuvent ainsi gérer cette problématique en prenant en charge la vérification de l'accès.

Les métadonnées inter aident les utilisateurs à trouver des ensembles de données pertinents qui peuvent répondre à leurs besoins. Elles facilitent ainsi pour l'utilisateur le travail de recherche de données au sein du lac. À nouveau, les métadonnées inter sont composées de plusieurs éléments :

- La **contenance** d'un jeu de données (*dataset containment*) permet de savoir qu'un ensemble de données est contenu dans un ou plusieurs ensemble(s) de données.
- Le **recouvrement partiel** (*partial overlap*) indique que des données peuvent partager des éléments en commun. Par exemple, des tables issues de bases de données relationnelles différentes peuvent contenir des attributs identiques.
- La **provenance** signifie qu'un ensemble de données est la source d'un autre ensemble de données.
- Un **groupe logique** (*logical cluster*) permet de savoir que des jeux de données sont dans le même domaine (par exemple, différentes versions d'un même jeu de données).
- La **similarité de contenu** indique que différents ensembles de données partagent les mêmes attributs.

La figure 2.7 résume cette catégorisation des métadonnées en intra et inter.

2.3.3 Métadonnées spécifiques selon le format des données

Avec un ensemble de métadonnées universelles qui peuvent être construites pour toutes les données ingérées dans le lac de données, nous disposons d'un socle d'informations permettant de décrire les données. Toutefois, il est souvent nécessaire d'entrer plus en détail dans la description des données, mais puisque celles-ci peuvent avoir des

Métadonnées intra

- Caractéristiques de données
- Métadonnées définitionnelles
- Métadonnées de navigation
- Lignage
- Métadonnées d'accès
- Métadonnées de qualité
- Métadonnées de sécurité

Métadonnées inter

- Contenance
- Recouvrement partiel
- Provenance
- Groupe logique
- Similarité de contenu

FIGURE 2.7 – Typologie de métadonnées intra - inter de [Ravat and Zhao, 2019b]

formats différents, des métadonnées spécifiques peuvent être nécessaires. Nous étudions ici des travaux qui ont été menés pour la construction de métadonnées spécifiques selon le type de données.

En ce qui concerne les données structurées, Farrugia et al. ont proposé d'extraire le schéma des données ainsi que les colonnes qui composent les données [Farrugia et al., 2016]. Ensuite, en croisant ces informations, les auteurs créent un lien entre deux tables si celles-ci ont au moins une colonne en commun. Toutefois, il n'est pas explicitement précisé quels critères permettent de déterminer si deux tables ont une colonne en commun, nous supposons donc que la comparaison est basée exclusivement sur l'intitulé des colonnes.

Sawadogo et al. ont mené des travaux sur les données textuelles dans les lacs de données [Sawadogo et al., 2019a]. Pour enrichir les métadonnées, les auteurs sont amenés à extraire d'un document textuel un sac de mots ou un vecteur de fréquence de termes (*term-frequency vector*), par exemple. Ces informations sont ensuite exploitées pour créer notamment des liaisons de similarité entre documents, en utilisant une métrique comme la similarité Cosinus [Allan et al., 2000], par exemple. Ces liaisons de similarité constituent des métadonnées.

Guidice et al. [Giudice et al., 2018, Diamantini et al., 2018] ont eux aussi travaillé sur les données non structurées dans les lacs de données. Leur approche se base sur la proximité des mots-clés associés aux données non structurées, et repose sur la théorie des graphes. Dans un premier temps, les mots-clés correspondent chacun à un nœud, et sont reliés à un nœud spécifique qui matérialise le document non structuré (par exemple une vidéo). Ensuite, à l'aide d'un thésaurus (BabelNet), les mots ayant un lien selon celui-ci sont reliés par un arc. Enfin, les mots-clés sont analysés deux à deux dans le but de chercher une similarité textuelle entre eux : si la similarité est suffisamment élevée, un autre arc est créé pour relier les nœuds correspondants. Toutes ces informations sont enregistrées dans le système de métadonnées.

Dans un contexte bien différent des lacs de données, de nombreux travaux sont menés sur le traitement des données audio [Schneider et al., 2019]. Schneider et al. proposent par exemple wav2vec, un modèle de reconnaissance de la parole, basé sur des méthodes d'apprentissage non supervisé. Les travaux sont orientés pour le traitement de la parole, mais la méthode est toutefois généralisable à n'importe quel contenu audio. Ainsi, pour

tout fichier audio, wav2vec génère un vecteur qui décrit le contenu audio de manière générique. Ces vecteurs peuvent ensuite être exploités pour comparer les fichiers audio entre eux. Ces travaux sont un exemple d'un processus qui pourrait être mis en œuvre dans un lac de données, pour extraire des informations spécifiques aux données audio lors de l'ingestion.

À la lumière de ces éléments, nous pouvons généraliser ces métadonnées spécifiques comme étant des descripteurs (ou *features*) qui permettent, comme le nom l'indique, de décrire au mieux les données. Ainsi, les descripteurs de données structurées sont essentiellement des informations portant sur les colonnes des données, tandis que ceux de données textuelles seront des sacs de mots ou des vecteurs de fréquence de termes. Nous avons alors, pour chaque ensemble de données présent dans le lac, à la fois des métadonnées universelles et des descripteurs spécifiques selon le format des données.

Le besoin de métadonnées spécifiques est évident, et a été soulevé par les auteurs des différents articles présentés plus haut. Plus le système de métadonnées du lac de données est riche et complet, plus ce dernier s'éloigne du marécage de données et est donc utilisable et exploitable. Cependant, ces descripteurs spécifiques peuvent être difficiles à gérer de manière uniforme, car sont souvent adaptés pour un cas d'usage spécifique. Il n'existe pas à notre connaissance de travaux sur une gestion unique de descripteurs différents au sein d'un système de métadonnées.

Nous proposerons dans la suite de ce manuscrit des descripteurs spécifiques pour les données structurées, ainsi qu'un assistant pour aider l'utilisateur dans la démarche de création de ces métadonnées spécifiques.

2.4 Organisation des métadonnées

La section précédente de cet état de l'art nous a permis de mettre en évidence le fait que les métadonnées d'un lac de données sont essentielles à son bon fonctionnement [Miloslavskaya and Tolstoy, 2016]. Seulement, en plus de définir la constitution des métadonnées, il faut aussi les organiser au sein d'un système de métadonnées, et c'est pourquoi plusieurs équipes de recherche ont travaillé sur cette problématique [Halevy et al., 2016, Hai et al., 2016, Diamantini et al., 2018, Ravat and Zhao, 2019b]. Il n'existe toutefois pas de consensus en ce qui concerne l'organisation des métadonnées, et c'est pourquoi les propositions sont assez variées : certaines sont des implémentations de lacs de données généralement peu détaillées et dédiées à un cas d'usage spécifique, tandis que d'autres sont des modèles de métadonnées, plus explicites, mais manquant de généricité pour s'adapter à des problèmes métiers variés.

Nous commençons par passer en revue l'état de l'art des systèmes de métadonnées de lacs de données dans la section 2.4.1, en proposant notamment six fonctionnalités pour évaluer la robustesse d'un système de métadonnées. À l'aide de ces fonctionnalités, nous comparons différents systèmes de métadonnées de la littérature. Ensuite, nous nous penchons plus particulièrement sur les modèles de métadonnées, qui sont explicites et réutilisables par rapport à des implémentations de lacs de données qui sont des « boites

noires » ne donnant pas assez de détails pour les réutiliser convenablement. L'étude des modèles de métadonnées est faite dans la section 2.4.2.

2.4.1 Systèmes de métadonnées

La gestion des métadonnées joue un rôle essentiel dans les lacs de données, car en l'absence d'un schéma fixe, l'interrogation et l'analyse des données dépendent d'un système de métadonnées efficace. Cependant, il n'existe pas à notre connaissance de critères objectifs de mesure de l'efficacité d'un système de métadonnées dans le contexte des lacs de données. C'est pourquoi nous proposons, d'une part, une liste de fonctionnalités clés attendues idéalement du système de métadonnées d'un lac de données (section 2.4.1.1) et, d'autre part, une comparaison de plusieurs systèmes de métadonnées sur la base de ces fonctionnalités (section 2.4.1.2). Cette comparaison se termine par une discussion sur les limites des systèmes de métadonnées qui ne sont pas des modèles de métadonnées, c'est-à-dire des systèmes conceptuels d'organisation des métadonnées, plus explicites et réutilisables.

2.4.1.1 Fonctionnalités de base d'un système de métadonnées

Nous identifions dans la littérature six fonctionnalités principales que devrait idéalement proposer le système de métadonnées d'un lac de données.

L'**enrichissement sémantique** (ES), aussi appelé annotation sémantique [Hai et al., 2016] ou profilage sémantique [Ansari et al., 2018], consiste à générer une description du contexte des données (avec des *tags*, par exemple) pour les rendre plus interprétables et compréhensibles [Terrizzano et al., 2015]. Il se fait à l'aide de bases de connaissances telles que des ontologies. L'annotation sémantique joue un rôle clé dans l'exploitation des données, dans le sens où elle permet de résumer les ensembles de données contenus dans le lac de sorte qu'ils soient plus compréhensibles par l'utilisateur. Elle peut également servir de base à l'identification de liaisons entre les données.

L'**indexation des données** (ID) consiste à mettre en place une structure de données pour retrouver des ensembles de données sur la base de caractéristiques précises (mots-clés ou motifs). Cela passe par la construction d'index ou d'index inversés. L'indexation permet d'optimiser l'interrogation des données dans le lac à travers un filtrage par mots-clés. Elle est particulièrement utile pour la gestion de données textuelles, mais peut également servir dans un contexte de données semi-structurées ou structurées, notamment pour des raisons d'efficacité [Singh et al., 2016, Haste, 2017].

La **génération et conservation de liaisons** (GL) consiste pour le système de métadonnées à détecter des relations de similarité, ou à intégrer des liaisons préexistantes entre des ensembles de données. L'intégration des liaisons entre les données peut servir à élargir la panoplie d'analyses possibles à partir du lac par la recommandation de données connexes à celles intéressant l'utilisateur [Maccioni and Torlone, 2018]. Les liaisons entre les données peuvent également servir à identifier des *clusters* de données, c'est-à-dire des regroupements de données fortement liées entre elles et différentes des autres [Farrugia et al., 2016].

Nous définissons le **polymorphisme des données** (PD) comme le fait d'avoir dans le système de métadonnées plusieurs représentations des mêmes données. Chaque représentation correspond alors aux mêmes données modifiées ou reformatées pour un besoin spécifique. Un document textuel peut par exemple être représenté sans mots vides (*stop-words*), sous forme de sac de mots, etc. Il est indispensable dans le contexte des lacs de données de structurer au moins partiellement les données non structurées afin de permettre leur analyse automatisée [Diamantini et al., 2018]. Ainsi, nous considérons comme [Stefanowski et al., 2017] qu'il est important d'avoir des modèles de métadonnées permettant de conserver simultanément plusieurs représentations des mêmes données afin d'éviter la répétition de certains prétraitements, et donc d'optimiser les analyses depuis le lac.

La fonctionnalité de **versionnement des données** (VD) désigne la faculté du système de métadonnées à prendre en charge les évolutions des données tout en conservant leurs états précédents. Autrement dit, le versionnement consiste à conserver plusieurs états du même ensemble de données dans le temps. Cette faculté est primordiale dans le contexte des lacs de données, car elle permet d'assurer la reproductibilité des analyses et de supporter la détection et la correction d'éventuelles erreurs ou incohérences. Le versionnement permet également de prendre en charge une évolution ramifiée des données, notamment dans leur schéma [Hellerstein et al., 2017].

Le **suivi d'utilisation** (SU) trace les interactions entre les utilisateurs du lac et les données. Ces interactions sont généralement des opérations de création, de modification ou de lecture des données. L'intégration de ces informations dans le système de métadonnées permet de comprendre et d'expliquer d'éventuelles incohérences dans les données [Beheshti et al., 2017]. Elles peuvent également servir à la gestion de données sensibles, par la détection d'intrusions [Suriarachchi and Plale, 2016].

Le suivi d'utilisation et le versionnement des données sont étroitement liés puisque les interactions induisent dans certains cas la création de nouvelles versions ou représentations des données. Cependant, ces fonctionnalités ne sont pas pour autant systématiquement proposées ensemble [Suriarachchi and Plale, 2016, Beheshti et al., 2017, Diamantini et al., 2018].

Pour récapituler, voici la liste des six fonctionnalités identifiées :

1. Enrichissement sémantique (ES)
2. Indexation des données (ID)
3. Génération et conservation de liaisons (GL)
4. Polymorphisme des données (PD)
5. Versionnement des données (VD)
6. Suivi d'utilisation (SU)

2.4.1.2 Comparaison des systèmes de métadonnées

Les six fonctionnalités identifiées dans la section précédente nous permettent d'évaluer la robustesse du système de métadonnées d'un lac de données, et sa capacité à gérer tous

les problèmes inhérents à la gestion des données au sein d'un lac. Nous proposons dans cette partie une comparaison de plusieurs systèmes de métadonnées sur la base de ces fonctionnalités.

Les systèmes de métadonnées considérés dans cette comparaison sont relatifs à deux types de travaux : les modèles de métadonnées d'une part, et les implémentations de lacs de données d'autre part. L'intitulé « modèles de métadonnées » désigne des systèmes conceptuels d'organisation des métadonnées dans les lacs. Ils ont l'avantage d'être plus détaillés et plus facilement reproductibles que les implémentations de lacs de données, qui se situent elles à un niveau plus opérationnel. Ces dernières sont des exemples de mise en œuvre de lacs de données pour lesquels le fonctionnement et les fonctionnalités résultants sont décrits, avec peu de détails sur l'organisation conceptuelle des métadonnées.

Nous incluons dans cette étude des systèmes (modèles ou implémentations) non explicitement associés au concept de lacs de données par leurs auteurs, mais dont les caractéristiques permettent de les y assimiler. C'est notamment le cas du modèle de métadonnées Ground [Hellerstein et al., 2017], qui peut très bien servir à organiser le système de métadonnées d'un lac de données.

Il est important de noter que cette étude a été effectuée il y a quelques temps, et que les modèles de métadonnées les plus récents (que nous présentons plus loin dans ce chapitre) ne sont pas pris en compte. Nous faisons le choix de ne pas mettre à jour cette étude pour rester fidèle aux constats qu'elle nous avait permis de faire et qui ont motivé nos travaux. Dans le chapitre 3, nous discuterons justement de cette étude à la lumière de travaux plus récents.

La comparaison de 15 systèmes de métadonnées de lacs de données (et assimilés) est présentée dans le Tableau 2.2. Ce tableau montre que les systèmes les plus complets en termes de fonctionnalités sont les lacs de données GOODS et CoreKG, avec cinq fonctionnalités proposées sur six. Ces systèmes se distinguent des autres par la prise en charge du polymorphisme et du versionnement des données. Il convient toutefois de noter que ces deux systèmes de métadonnées sont des « boites noires » présentant peu de détails sur l'organisation conceptuelle des données. Le modèle de données Ground peut donc leur être préféré car il est beaucoup plus détaillé et presqu'aussi complet, en proposant quatre fonctionnalités sur les six identifiées.

Sur le plan des fonctionnalités, nous constatons une quasi-unanimité sur la pertinence de l'enrichissement sémantique avec 12 systèmes sur 15 proposant cette fonctionnalité et, dans une moindre mesure, les fonctionnalités d'indexation des données (9/15) et de génération de liaisons entre les données (8/15). En revanche, d'autres fonctionnalités sont beaucoup moins partagées, en particulier le polymorphisme (4/15) et le versionnement (3/15) des données. À notre sens, cette rareté ne dénote pas pour autant un manque de pertinence, mais plutôt une complexité de mise en œuvre. En effet, ces fonctionnalités se trouvent principalement dans les systèmes les plus complets (GOODS, CoreKG et Ground), et peuvent donc être considérées comme avancées, à l'inverse des autres fonctionnalités qui sont plus usuelles.

Soulignons toutefois que nous prenons ici un critère simple, voire réducteur, pour com-

parer les systèmes de métadonnées entre eux, puisque nous nous contentons de compter le nombre de fonctionnalités prises en charge. Une comparaison plus poussée, par exemple en pondérant les fonctionnalités selon leur importance, pourrait être intéressante à mener pour affiner les résultats obtenus.

Système \ Fonctionnalités	**Type**	**ES**	**ID**	**GL**	**PD**	**VD**	**SU**	**Total**
SPAR [Fauduet and Peyrard, 2010]	♦♯	✓	✓	✓			✓	4/6
[Alrehamy and Walker, 2015]	♦	✓		✓				2/6
[Terrizzano et al., 2015]	♦	✓	✓			✓	✓	4/6
Constance [Hai et al., 2016]	♦	✓	✓					2/6
GEMMS [Quix et al., 2016]	◊	✓						1/6
CLAMS [Farid et al., 2016]	♦	✓						1/6
[Suriarachchi and Plale, 2016]	◊				✓		✓	2/6
[Singh et al., 2016]	♦	✓	✓	✓	✓			4/6
[Farrugia et al., 2016]	♦			✓				1/6
GOODS [Halevy et al., 2016]	♦	✓	✓	✓		✓	✓	5/6
CoreDB [Beheshti et al., 2017]	♦		✓				✓	2/6
Ground [Hellerstein et al., 2017]	◊♯	✓	✓			✓	✓	4/6
KAYAK [Maccioni and Torlone, 2018]	♦	✓	✓	✓				3/6
CoreKG [Beheshti et al., 2018]	♦	✓	✓	✓	✓		✓	5/6
[Diamantini et al., 2018]	◊	✓		✓	✓			3/6
Total		12/15	9/15	8/15	4/15	3/15	7/15	

♦ : Implémentation d'un lac de données ◊ : Modèle de métadonnées
♯ : Modèle ou implémentation assimilable à un lac de données

Table 2.2 – Fonctionnalités proposées par les systèmes de métadonnées de lacs de données

Les implémentations sont des mises en œuvre technique d'un système de métadonnées. Elles sont généralement spécifiques pour un cas d'usage donné, et les auteurs ont effectué des choix techniques qui, souvent, ne permettent pas la réutilisation de l'implémentation sur un cas d'usage différent. De plus, il est assez fréquent que les implémentations soient des « boites noires », c'est-à-dire que les auteurs donnent peu de détails sur l'organisation conceptuelle des métadonnées. À l'inverse, un modèle de métadonnées fait référence à un système conceptuel d'organisation des métadonnées, qui explicite donc clairement son fonctionnement. Ceci le rend alors plus facilement réutilisable sur des cas d'usage différents.

Tous ces éléments nous poussent à considérer que les implémentations de lacs de

données ne permettent pas de proposer une solution viable concernant l'organisation des métadonnées. Nous préférons donc nous focaliser sur les modèles de métadonnées.

2.4.2 Modèles de métadonnées

Parmi les approches permettant de gérer les métadonnées dans les lacs de données, seuls les modèles de métadonnées, qui proposent une organisation conceptuelle des métadonnées, fournissent suffisamment de détails pour garantir leur réutilisation. Les modèles de métadonnées se trouvent donc être plus génériques que les implémentations, et doivent permettre de s'adapter à des cas d'usage différents. Mais les modèles doivent aussi être implémentables, idéalement de plusieurs façons, par exemple en utilisant des technologies différentes.

Les premiers modèles de métadonnées pour lacs de données proposés dans la littérature se voulaient génériques, mais il est clair aujourd'hui que cet objectif n'a pas été pleinement atteint. Par conséquent, nous résumons brièvement les premiers modèles de métadonnées (section 2.4.2.1), puis nous détaillons les plus récents et les plus exhaustifs (sections 2.4.2.2 et 2.4.2.3).

2.4.2.1 Premiers modèles de métadonnées

The Generic and Extensible Metadata Management System (GEMMS) est un pionnier des modèles de métadonnées génériques pour les lacs de données [Quix et al., 2016]. GEMMS comporte deux entités abstraites : fichier de données et unité de données. Un fichier de données représente une source de données, et une unité de données représente un élément de données identifiable dans une source de données. Chaque fichier de données est composé d'un ensemble d'unités de données, par exemple, un tableur est composé d'un ensemble de feuilles. Les fichiers de données et les unités de données peuvent être enrichis de valeurs de métadonnées atomiques ou complexes. Cependant, GEMMS nécessite des informations sur la structure des données pour fonctionner, ce qui le rend peu pratique lorsqu'il faut travailler avec des données non structurées.

Bien qu'il ne soit pas spécifiquement conçu pour les lacs de données, Ground est un autre modèle de métadonnées générique [Hellerstein et al., 2017]. Ground suit le contexte des données (métadonnées) à trois niveaux : 1) propriétés des métadonnées, 2) historique d'utilisation des données et 3) versionnement des données. Bien que plus étendu que GEMMS, Ground, comme GEMMS, ne prend pas en charge la liaison des données, alors que ce type de métadonnées a été identifié comme pertinent dans les lacs de données [Sawadogo et al., 2019b, Farrugia et al., 2016].

Enfin, basé sur le principe de composition des données similaire aux concepts de fichier de données et unité de données de GEMMS, le modèle de Diamantini et al. ajoute des liens de similarité entre les unités de données pour relier indirectement les fichiers de données [Diamantini et al., 2018]. Cependant, ce modèle n'inclut pas des métadonnées importantes telles que le versionnement des données ou le suivi d'utilisation de Ground.

2.4.2.2 DAMMS

Ravat et Zhao proposent un modèle de métadonnées intitulé DAMMS (*DAta lake Metadata Management System*). Comme dans le modèle de Diamantini et al., chaque fichier de données peut être associé à des métadonnées atomiques et complexes : propriétés des métadonnées, historique des données et liens avec d'autres fichiers de données. La principale contribution de DAMMS est d'associer la gestion des métadonnées à une architecture de zone. De nombreux lacs de données sont en effet architecturés en zones, avec par exemple une zone de données brutes et une zone de données traitées [Ravat and Zhao, 2019b, Giebler et al., 2019].

Plus concrètement, DAMMS représente les données à travers le concept de *dataset* (jeu de données) et les spécialisations des ensembles de données : *source dataset* pour les données entrantes, *data lake dataset* pour les données du lac, *semi/unstructured dataset* pour les données non structurées ou semi-structurées, et *structured datasets* pour les données structurées.

Le modèle comprend également le concept de relation qui définit les connexions entre les données, et les mots-clés qui permettent d'associer des *tags* aux ensembles de données. Enfin, les interactions entre les entités (utilisateurs et applications) et les données du lac sont tracées à l'aide des concepts accès, ingestion et traitement, qui correspondent respectivement aux opérations de lecture, d'intégration et de transformation des données.

Au-delà de la conformité avec les architectures de zone, DAMMS permet également une implémentation facile. En effet, le modèle est basé sur des concepts concrets que les utilisateurs peuvent facilement faire correspondre à leur cas d'utilisation. Cependant, cet avantage induit un manque de flexibilité dans l'utilisation du modèle.

La figure 2.8 présente un diagramme de classes UML qui représente les concepts du modèle de métadonnées DAMMS. Notons ici qu'un losange représente une relation n-aire entre des classes.

2.4.2.3 HANDLE

HANDLE (*Handling metAdata maNagement in Data LakEs*) vise à introduire plus de flexibilité que les modèles précédents, c'est-à-dire à prendre en charge un maximum de caractéristiques de modélisation des métadonnées à travers un nombre minimal de concepts [Eichler et al., 2020]. Les concepts de HANDLE sont subdivisés en un modèle de base et des extensions [Eichler et al., 2021].

Les concepts du modèle de base sont les entités de données et entités de métadonnées. Les entités de données représentent des fichiers de données et des parties de fichiers de données, de manière indistincte. Par exemple, une entité de données peut représenter une table relationnelle, tandis qu'une autre représente un n-uplet. Une entité de métadonnées est utilisée pour associer un ensemble de descriptions à une entité de données.

HANDLE propose deux types de relations entre les entités de données. Les *relations de similarité* relient des entités de données de même niveau de granularité, tandis que les *relations de contenance* expriment une relation hiérarchique.

À nouveau, nous donnons un diagramme de classes UML dans la figure 2.9 pour

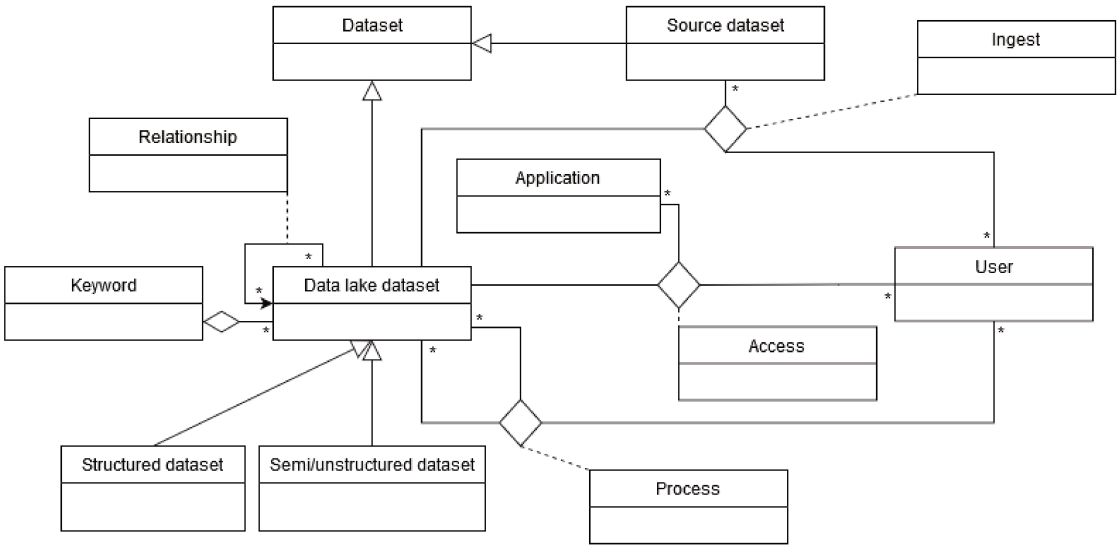

FIGURE 2.8 – Diagramme de classes UML du modèle conceptuel de DAMMS [Ravat and Zhao, 2019b]

représenter de manière visuelle le modèle conceptuel de HANDLE. Il est très clair, à la lecture de ce diagramme, que le concept d'entité de données (*Data*) est central, et que les autres concepts du modèle gravitent autour.

Trois extensions complètent le modèle par des métadonnées particulières. L'*indicateur de granularité* est associé à une entité de données. De même, l'*indicateur de zone* représente le niveau de raffinement des données, par exemple, des données brutes, des données prétraitées, etc. Enfin, le concept de *catégorisation* permet de définir plusieurs catégories d'entités de métadonnées.

En somme, HANDLE adopte un niveau d'abstraction élevé qui se conforme à la plupart des cas d'utilisation de la modélisation des métadonnées. Il peut notamment organiser les données en zones comme DAMMS, tout en gérant simultanément plusieurs niveaux de granularité comme GEMMS. Cependant, le niveau d'abstraction de HANDLE pourrait ne pas être suffisant, puisque des extensions au modèle de base sont nécessaires. Ainsi, comme dans le modèle de métadonnées précédent, si certains concepts qui ne sont pas déjà présents dans HANDLE sont nécessaires, de nouvelles extensions doivent être conçues.

2.5 Architectures et technologies de lacs de données

En complément des travaux portant sur les systèmes de métadonnées présentés dans la section précédente, d'autres recherches ont été menées sur les architectures des lacs de données, pour définir leur structure générale ainsi que les principaux composants [Sawadogo and Darmont, 2021, Ravat and Zhao, 2019a, Maccioni and Torlone, 2017, Hai et al., 2016]. Puisque la définition du concept de lac de données n'a pas été totalement

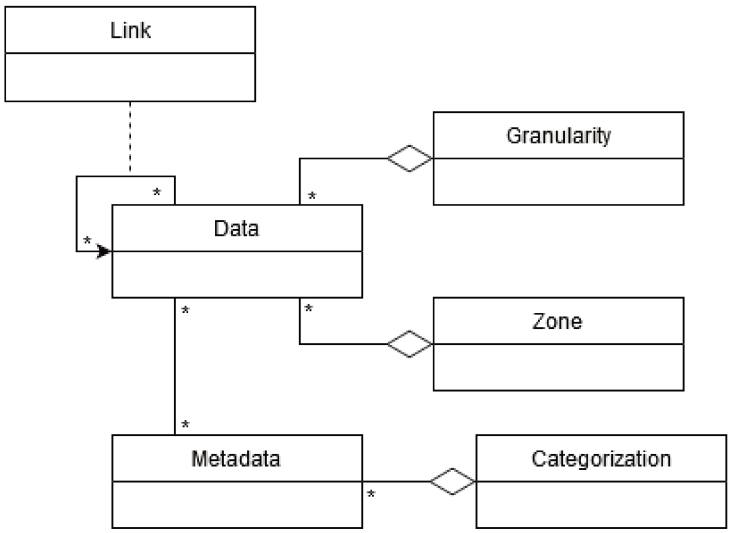

FIGURE 2.9 – Diagramme de classes UML du modèle conceptuel de HANDLE (base) [Eichler et al., 2020]

consensuelle par le passé, et que des propositions d'architecture ont émergé en parallèle, il existe plusieurs visions [Hellerstein et al., 2017, Halevy et al., 2016], que nous allons détailler. En outre, il existe de nombreux outils et technologies permettant d'implémenter ces architectures de lacs de données [Sawadogo and Darmont, 2021, Inmon, 2016, Miloslavskaya and Tolstoy, 2016]. Il est d'ailleurs intéressant de noter que certaines technologies ont émergé avant la première mention du nom de lac de données par Dixon en 2010, et donc que des outils ne sont pas directement lié au concept de lac, même si elles sont tout de même utilisables pour en implémenter.

Nous débutons cette section en présentant dans la section 2.5.1 les différentes architectures de lacs de données qui existent dans la littérature. La section 2.5.2 présente ensuite les différents outils et technologies permettant de mettre en œuvre de manière concrète un lac de données.

2.5.1 Architectures de lacs de données

Nous distinguons trois approches majoritaires pour définir l'architecture d'un lac de données : l'organisation selon le type des données (bassins), selon la maturité des données (zones), ou selon la sémantique des données (objets). Nous détaillons ces différentes approches, pour finir par mener une discussion sur celles-ci.

2.5.1.1 Stockage de données volumineuses

Parmi les propositions d'architectures, certains considèrent le lac de données uniquement comme un espace de stockage économique pour des données massives [Miloslavskaya and Tolstoy, 2016]. Dans cette vision, les propositions d'architecture sont étroitement liées

à des technologies, voire même servant uniquement de motivation pour promouvoir une technologie, comme relevé par [Fang, 2015].

Cette manière de faire est intimement liée aux premières définitions du concept de lac de données, basées essentiellement sur le stockage de données volumineuses. Il est intéressant de noter que cette vision est parfois encore d'actualité, pour justifier l'émergence de nouveaux termes comme *lakehouse* ou *data mesh* (tous deux présentés dans la section 2.2.3).

Pour ces raisons, nous préférons mettre de côté cette vision car elle ne correspond pas à notre définition d'un lac de données.

2.5.1.2 Bassins de données

Une première architecture, introduite initialement par Inmon, propose une organisation en bassins de données (*data ponds*) [Inmon, 2016]. L'idée de cette approche est de partitionner les données en trois bassins en fonction de leur format. Dans l'architecture d'Inmon, les données véloces, i.e. qui sont produites à haute fréquence, vont dans le bassin des données analogiques (*analog data pond*). Ce sont généralement des données en provenance de l'internet des objets (IoT). Les données provenant d'applications, qui sont essentiellement des données structurées issues de bases de données relationnelles, sont enregistrées dans le bassin des données d'application (*application data pond*). Il est intéressant de noter qu'Inmon considère que ce bassin est en fait un entrepôt de données. Finalement, les données textuelles non structurées sont aiguillées vers le bassin de données textuelles (*textual data pond*).

Chaque bassin possède ses propres outils pour le stockage et le traitement des données. À cela, Inmon ajoute deux autres bassins spécifiques : en amont, un bassin de données brutes, qui sert de zone de transit où les données nouvellement ingérées sont préparées pour être envoyées dans les bassins adéquats. En aval, nous retrouvons un bassin de données d'archives, dans lequel des données issues des trois bassins qui sont très peu utilisées y sont déposées, pour les conserver en cas de besoin futur.

La figure 2.10 illustre le fonctionnement de cette architecture, avec les cinq bassins de données. Les données arrivent dans le bassin de données brutes, puis sont dispatchées dans les trois bassins (données analogiques, données d'application, données textuelles) selon leur type. Enfin, ces trois mêmes bassins peuvent envoyer des données dans le bassin d'archive si elles sont peu exploitées. Précisons que les trois bassins principaux (données analogiques, données d'application, données textuelles) proposent chacun les outils de traitement et d'exploitation des données qu'ils contiennent.

Cette architecture de lac de données présente des similarités avec l'architecture Lambda [Gröger, 2018]. John et Misra proposent un lac de données qui prend appui sur cette architecture [John and Misra, 2017]. Dans leur architecture de lac, les données sont à nouveau distinguées selon leur nature, mais cette fois par rapport leur vélocité. Il existe une couche pour les données de flux (*streaming data*), à savoir les données qui sont générées très rapidement, voire en temps réel, et dont l'exploitation est spécifique. Une seconde couche existe pour les données par lots (*bulk data*), c'est-à-dire celles qui sont

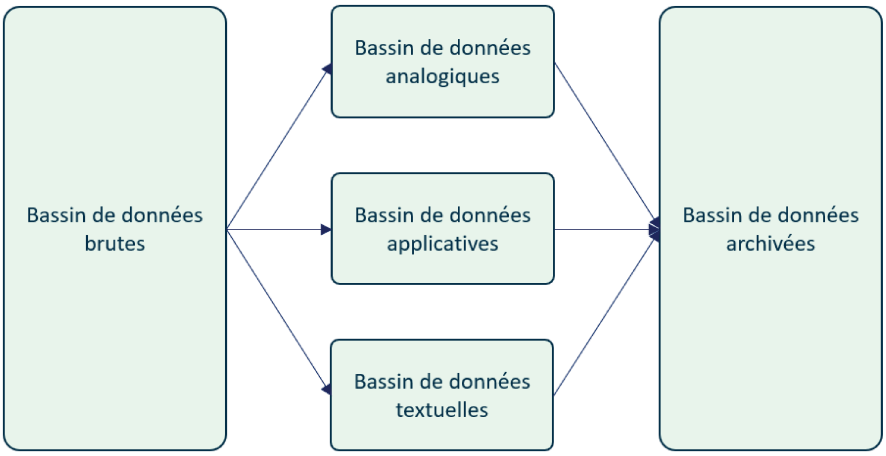

FIGURE 2.10 – Architecture basée sur les bassins de données [Inmon, 2016]

générées à une vitesse plus raisonnable, voire de manière ponctuelle. Ces données sont généralement traitées par des méthodes plus traditionnelles.

En amont de ces deux couches se trouvent divers mécanismes pour récupérer les données. Les données sont extraites de leurs sources dans la couche d'acquisition, puis une couche de messagerie sert à les tracer. Ensuite, la couche d'intégration pré-traite les données avant de les envoyer vers le composant adéquat, à savoir pour les données de flux ou les données par lots. En aval, nous trouvons une couche de service qui met à disposition les données présentes dans les deux couches de stockage.

Nous représentons graphiquement l'architecture Lambda dans la figure 2.11. Nous constatons les différentes couches, respectivement d'acquisition, de messagerie et d'intégration, qui permettent d'acheminer les données vers leur zone de stockage en fonction de la nature (très véloces ou non). En bout de chaîne, la couche de service met à disposition les données enregistrées dans le lac.

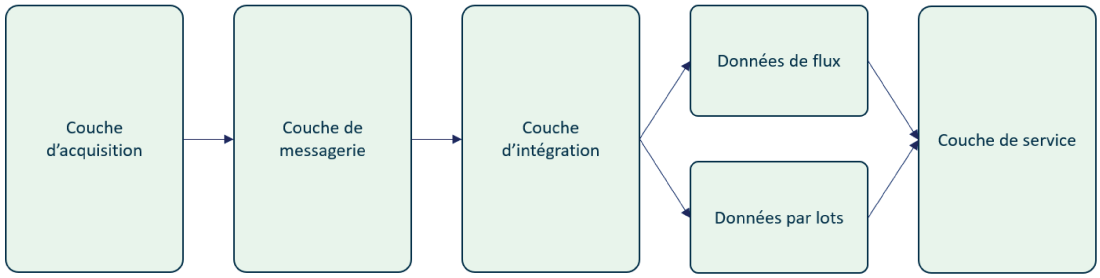

FIGURE 2.11 – Architecture Lambda [John and Misra, 2017]

Ces architectures de lacs de données présentent différents espaces de stockage dans lesquels les données sont entreposées. Même si leur nombre varie, ces espaces de stockages séparent les données selon leur format. Aussi, l'ingestion des données dans le lac

est effectuée de manière uniforme (bien qu'elles soient ensuite envoyées dans les espaces de stockage adéquats). En revanche, l'exploitation des données du lac diffère selon l'architecture : dans un cas, une couche d'accès sert à accéder aux données par lots ou aux données de flux [John and Misra, 2017], alors que dans l'autre cas, chaque bassin possède des propres outils d'exploitation de données [Inmon, 2016].

2.5.1.3 Zones de données

D'autres architectures de lacs de données préfèrent organiser le lac en zones, où les zones font référence au « degré de raffinement » des données, c'est-à-dire leur niveau de maturité [Giebler et al., 2019].

LaPlante et Sharma proposent une architecture de lac de données basée sur 7 zones [LaPlante and Sharma, 2016]. La première est la zone de transit des données qui permet de récupérer les données à ingérer dans le lac. Ces données, dans leur format natif, sont envoyées dans la zone de données brutes pour y être stockées. Ensuite, une zone de raffinement permet de retravailler ces données brutes en fonction des besoins (nettoyage, agrégation, normalisation...). Une fois les données retravaillées et prêtes à être exploitées, elles sont déposées dans la zone des données fiables. La cinquième zone est le bac à sable, qui est destiné aux *data scientists* pour leur permettre de mener des analyses exploratoires et prédictives sur les données fiables. En bout de chaîne se trouve la zone de consommation, contenant des données qui sont utilisées régulièrement, et généralement par des utilisateurs métiers. Finalement, la zone de gouvernance constitue la dernière zone de cette architecture, et elle communique avec les six autres zones dans le but de gérer le cycle de vie des données, mais aussi leur qualité et leur sécurité.

L'architecture du lac de données est représentée dans la figure 2.12. Nous voyons l'enchaînement des six zones, dans lesquelles les données se décantent petit à petit, passant de données brutes à des données accessibles, fiables et exploitables. Ce raffinement progressif des données se fait à travers des étapes intermédiaires. Enfin, la septième zone est transverse et communique avec toutes les autres zones, pour suivre la gouvernance des données au fur et à mesure de leur évolution.

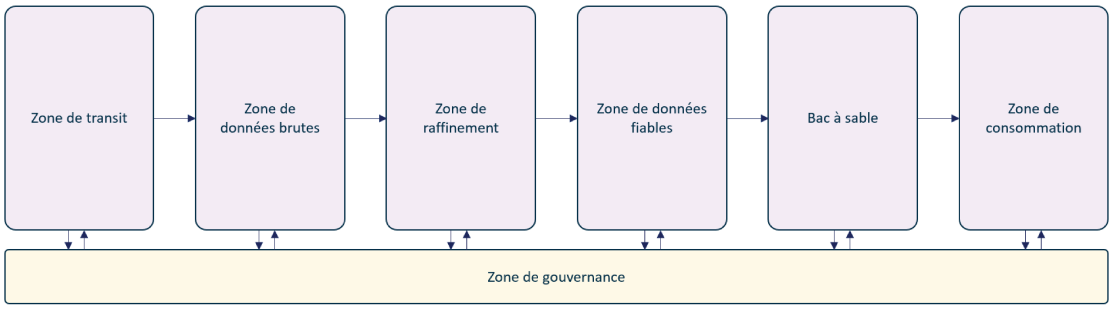

FIGURE 2.12 – Architecture basée sur les zones de données [LaPlante and Sharma, 2016]

Par la suite, Ravat et Zhao ont proposé eux aussi une architecture de lac de données en zones, en marge de leur modèle de métadonnées [Ravat and Zhao, 2019b]. Dans

leur architecture, les données brutes, donc telles qu'elles ont été ingérées dans le lac, sont enregistrées dans la zone des données brutes sans subir de modification (*raw data zone*). Ainsi, toutes les données présentes dans le lac passent par cette zone. Dans un second temps, si les données sont retravaillées, elles sont alors placées dans la zone de traitement (*processing zone*). Dans cette zone, les données peuvent être à des degrés de raffinement différents, selon la quantité de travail à fournir pour arriver au produit fini, i.e. les données totalement retravaillées. Enfin, les données prêtes à l'emploi sont placées dans la zone d'accès (*access zone*). Ces données sont généralement les plus exploitées par les utilisateurs du lac, et donc cette zone garantit leur maturité et leur accessibilité. Notons par ailleurs que d'autres auteurs qualifient ces trois zones de bronze, argent et or, respectivement [Heintz and Lee, 2019].

La différence majeure avec la précédente architecture en zones (le lac de données de [LaPlante and Sharma, 2016]) réside dans le fait qu'ici, chacune des trois zones possède, en plus d'un stockage de données, ses propres outils pour exploiter les données qu'elles contiennent. La zone de données brutes contient les outils pour l'ingestion, tandis que la zone de traitement possède de quoi transformer les données pour les raffiner. Finalement, la zone d'accès propose aux utilisateurs des méthodes pour interroger les données prêtes à l'emploi.

La figure 2.13 illustre un lac de données basé sur l'architecture de Ravat et Zhao, avec trois zones de données. Nous voyons que les données arrivent en premier lieu dans la zone des données brutes. Si elles sont retravaillées, elles passent alors dans la zone de traitement. Finalement, la zone d'accès contient les données les plus raffinées, prêtes à l'emploi. Notons aussi que chacune des trois zones possède ses propres espaces de stockages et outils d'exploitation de données.

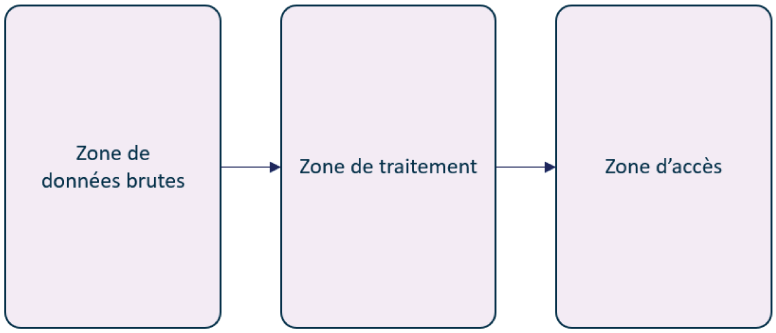

FIGURE 2.13 – Architecture basée sur les zones de données [Ravat and Zhao, 2019b]

À nouveau, ces architectures de lac de données reposent sur une distinction des données, cette fois-ci pas sur leur format, mais leur niveau de maturité. Le lac de données présente donc différents espaces de stockage de données pour compartimenter les données et éviter que des données brutes ne cohabitent avec des données raffinées, prêtes à l'emploi. Ici, les données ingérées dans le lac passent toutes par la zone de données brutes, puis sont amenées à être retravaillées dans les différentes zones en aval. Toutefois,

le lac de données de LaPlante et Sharma n'évoque pas de manière explicite les moyens de traitement des données dans les différentes zones [LaPlante and Sharma, 2016], tandis que Ravat et Zhao spécifient clairement que les trois zones possèdent leurs propres outils d'exploitation de données [Ravat and Zhao, 2019b].

2.5.1.4 Sémantique des données

La troisième famille d'architectures de lacs de données se base sur la sémantique des données, c'est-à-dire la signification de l'information portée par les données.

Diamantini et al. proposent, à l'aide de leur modèle de métadonnées, une architecture où les données sont regroupées en fonction de leur signification [Diamantini et al., 2018]. Les auteurs introduisent la notion d'objet, dont la définition dépend du type des données sources. Par exemple, dans une base de données relationnelle, un objet correspond à une table et ses attributs, tandis que dans un document XML ou JSON, un objet représente un élément simple ou complexe ainsi que ses attributs. De plus, les objets peuvent être liés entre eux, à travers par exemple des liens de similarité.

La force d'un objet est de pouvoir y associer des métadonnées, qui peuvent être assez riches d'un point de vue sémantique. Il est par exemple possible d'y associer des termes pour le décrire, et ces termes peuvent provenir d'une ontologie. À l'aide de celle-ci, il est possible de détecter si des termes associés à différents objets sont proches d'après l'ontologie, par exemple.

La figure 2.14 illustre un lac de données basé sur la sémantique avec quatre objets, représentés en bleu, chacun correspondant à un ensemble de données. Les objets identifiés 1 et 2 (resp. 3 et 4) sont connectés par un lien de similarité, dont la valeur est de 0,69 (resp. -0,17).

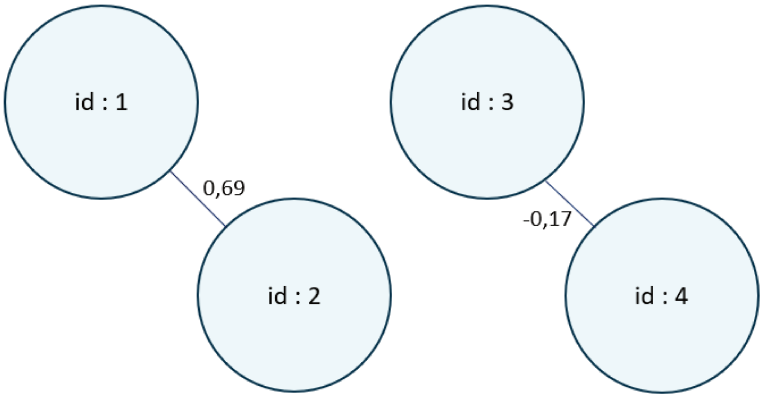

FIGURE 2.14 – Architecture basée sur la sémantique des données [Diamantini et al., 2018]

Baars et Ereth ont proposé une architecture pour répondre aux défis posés par l'exploitation des mégadonnées [Baars and Ereth, 2016]. Bien que ce ne soit pas explicitement présenté comme un lac de données, leur proposition partage beaucoup de points communs avec une architecture de lac de données basée sur la sémantique. Ils proposent

une plateforme *Business Intelligence & Analytics* qui s'appuie sur ce qu'ils appellent des atomes analytiques, qu'ils considèrent comme de petits entrepôts de données autonomes prêts à être analysés. Les données de l'atome sont historisées, versionnées, enrichies et nettoyées. Un second intérêt de cette proposition réside dans la combinaison d'atomes analytiques, pour former ce que les auteurs désignent comme des entrepôts de données virtuels.

L'atome analytique présente ici une forte similitude avec la notion d'objet de [Diamantini et al., 2018]. Un atome, tout comme un objet, est associé à un ensemble cohérent de données, et possède plusieurs métadonnées associées. Par exemple, les données sont enrichies, et il est possible d'y retrouver différentes versions des données. De plus, en assemblant des atomes analytiques entre eux, cela équivaut à créer des liens entre objets de données.

Une illustration du fonctionnement de cette architecture est présentée dans la figure 2.15. Sur cette figure tirée directement de leur proposition, nous voyons les atomes analytiques enregistrés dans la plateforme, qui peuvent ensuite être assemblés pour constituer des « *Virtual DW* », chaque entrepôt de données virtuel servant à répondre à un besoin d'analyse spécifique.

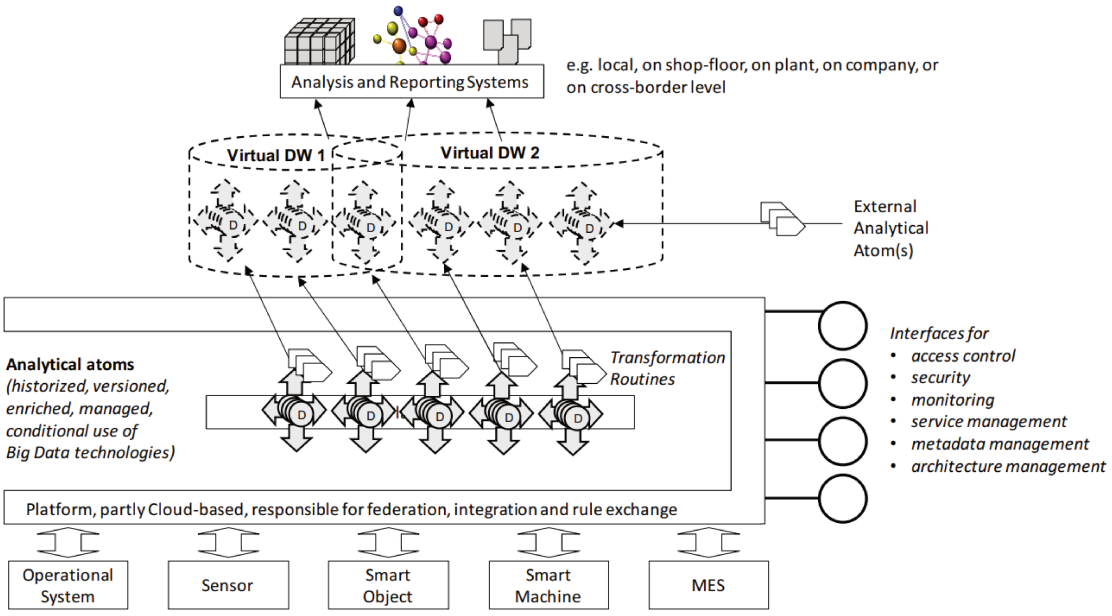

FIGURE 2.15 – Architecture basée sur les atomes analytiques (image tirée de [Baars and Ereth, 2016])

Cette architecture de lac de données diffère assez fondamentalement des deux premières architectures présentées plus haut, car il est ici impossible d'énoncer à l'avance le nombre d'objets qui composeront le lac de données. En effet, là où il est possible de délimiter les formats des données à compartimenter, ou le nombre de zones pour la maturité des données, il est toutefois impossible de prédire le nombre d'ensembles de

données qui seront ingérés dans le lac de données. Cette architecture présente un intérêt certain puisqu'elle offre une plus grande flexibilité. Toutefois, cette flexibilité s'accompagne d'une plus grande différence entre le niveau fonctionnel et l'implémentation, qui est plus dépendante du cas d'usage traité.

2.5.1.5 Discussion

Dans des travaux récents, Sawadogo et Darmont ont proposé une classification différente pour les architectures de lacs de données [Sawadogo and Darmont, 2021]. Au lieu de distinguer les architectures en bassins de données et en zones, les auteurs proposent trois catégories d'architecture :

- Les **architectures fonctionnelles** donnent les composants qui vont constituer un lac de données en se basant sur les fonctions qu'ils vont accomplir (ingestion, stockage des données, gestion des métadonnées, etc.).
- Les **architectures en zones** découpent les composants du lac de données en zones, en particulier selon la maturité des données.
- Les **architectures hybrides** définissent les composants du lac à la fois selon des zones, mais aussi selon des fonctions.

Avec cette classification, l'architecture Lambda [John and Misra, 2017] est considérée comme une architecture fonctionnelle, tandis que le lac de données de [LaPlante and Sharma, 2016] appartient à la catégorie des architectures en zones. Finalement, les architectures hybrides incluent les architectures de [Ravat and Zhao, 2019b] et [Inmon, 2016]. Notons que Sawadogo et Darmont considèrent les bassins de données d'Inmon comme des zones, bien qu'il ne soit pas question de maturité des données dans ce cas de figure.

Cette classification des architectures de lac de données présente un intérêt puisqu'elle permet de montrer que la frontière entre les architectures fonctionnelles et en zones est poreuse, et qu'à leur intersection se trouvent les architectures hybrides. Toutefois, cette classification n'inclut pas les architectures basées sur la sémantique des données, qui sont certes différentes des autres architectures de lacs de données, mais qui doivent tout de même être mentionnées selon nous.

Peu importe la catégorisation des architectures de lacs de données (architectures fonctionnelles, en zones et hybrides comme le proposent [Sawadogo and Darmont, 2021], ou bien architectures en bassins, en zones et en objets sémantiques), nous soulignons la pertinence des propositions de la littérature, et le fait que ces architectures sont adaptées pour répondre aux problèmes identifiés par leurs auteurs. Néanmoins, nous considérons que ces différentes architectures proposées ont une forte tendance à compartimenter les données du lac. En effet, nous avons présenté plus haut des architectures qui organisent le lac autour du type des données, de leur niveau de maturité, ou de leur signification sémantique. Nous pensons que cela est contraire à la propriété *schema-on-read* inhérente aux lacs de données.

Pour illustrer, considérons l'exemple suivant. Une entreprise disposant d'un grand nombre de documents textuels souhaite extraire des descripteurs structurés (par exemple,

un vecteur de fréquence de termes) pour ensuite mesurer la similarité entre les documents. À cette fin, les documents textuels et leurs descripteurs sont stockés dans un lac de données. Faisons la distinction entre les trois cas de figure énumérés ci-dessus.

Si le lac de données adopte une architecture basée sur la structure des données, et crée ainsi des bassins de données au sein du lac, alors les documents textuels se trouvent dans le bassin des données textuelles, tandis que les descripteurs sont enregistrés dans le bassin des données d'application. Avec une architecture prenant appui sur la maturité des données et créant des zones, les documents textuels se situent dans la zone des données brutes, et les descripteurs se trouvent dans une autre zone (par exemple, les données de traitement ou les données d'accès). Enfin, dans le cas où le lac est basé sur la sémantique des données, chaque document textuel crée un objet, qui contient à la fois les données brutes (donc le document textuel) ainsi que ses descripteurs.

Nous observons dans cet exemple que, selon l'architecture choisie pour le lac de données, les données sont placées à différents endroits dans le lac. Par conséquent, l'utilisation du lac sera différente selon son architecture. Nous pensons que ces propositions sont trop spécifiques et nous proposerons plus loin dans ce manuscrit une architecture plus générale qui évite de compartimenter les données du lac.

2.5.2 Technologies pour lac de données

Pour implémenter un lac de données, il existe de multiples possibilités, allant de la solution tout-en-un à l'assemblage d'outils indépendants, en passant par des éco-systèmes proposant un grand nombre de services. Nous passons en revue ces possibilités.

Nous commençons par lister dans la section 2.5.2.1 des technologies qui, si assemblées, peuvent permettre d'implémenter un lac de données. Dans la section 2.5.2.2, nous étudions ensuite des écosystèmes complets, qui proposent tout un panel d'outils et de technologies, en particulier celui proposé par Apache. Nous investiguons finalement les *clouds* propriétaires dans la section 2.5.2.3.

2.5.2.1 Assemblage de technologies autonomes

Il existe de nombreuses technologies autonomes, souvent ouvertes, qui peuvent être assemblées de manière intelligente pour former un lac de données [Gupta et al., 2017, Davoudian et al., 2018]. Ainsi, il est possible d'ingérer, de stocker et d'exploiter des mégadonnées, tout en disposant d'un système de métadonnées pour pouvoir s'y retrouver aisément [Beheshti et al., 2017, Khine and Wang, 2017, John and Misra, 2017].

Pour enregistrer les données, en particulier les données structurées, il existe les traditionnels systèmes de gestion de bases de données (SGBD) relationnels, qui permettent d'enregistrer les données sous formes de tables possédant des attributs. Plusieurs outils existent, qu'ils soient gratuits ou payants, et nous pouvons par exemple citer MySQL [2], PostgreSQL [3], ou encore Oracle Database [4].

2. `https://www.mysql.com/`
3. `https://www.postgresql.org/`
4. `https://www.oracle.com/database/`

Toutefois, ces SGBD « classiques » s'avèrent limités pour stocker des mégadonnées, d'une part en termes de volumétrie, à cause de leurs propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité) qui limitent leur capacité à passer à l'échelle [Krishnan, 2013]. C'est aussi la variété des mégadonnées qui pose problème aux SGBD classiques : en effet, il est très difficile voire impossible d'y stocker des données semi-structurées et non structurées. Ainsi, pour pallier ces limites, les SGBD NoSQL (*not only SQL*) ont fait leur apparition, et nous en distinguons quatre types [Pathirana, 2015, John and Misra, 2017].

- Les SGBD de type **clés-valeurs** associent des données, ou valeurs, à une clé. Ce mode de fonctionnement est assimilable à celui d'une table d'association. Par exemple, Redis [5] est un SGBD clés-valeurs.
- Les SGBD **orientés documents** peuvent enregistrer des documents qui n'ont pas de schéma prédéfini, comme du JSON. Ils sont donc particulièrement adaptés pour enregistrer des données semi-structurées. MongoDB [6] est un exemple de SGBD orienté documents.
- Les SGBD **orientés colonnes** permettent de contourner les problématiques de volumétrie pour les données structurées. Les données sont enregistrées sous formes de tables, mais ces tables sont des ensembles de colonnes, et non des ensembles de n-uplets comme dans les SGBD relationnels classiques. Un exemple de SGBD orienté colonnes disponible sur le marché est Hypertable [7].
- Le dernier type de SGBD NoSQL est celui des SGBD **graphes**, qui sont particulièrement adaptés pour mettre en avant les relations entre les données. Comme le nom l'indique, les données sont enregistrées sous forme de graphe, ce qui est bien adapté pour les données issues de réseaux sociaux. Neo4j [8] est un exemple de SGBD graphes.

En marge de l'entreposage des données, des outils et systèmes existent pour les traiter et les exploiter. Par exemple, ElasticSearch [9] permet d'indexer les données et ainsi pouvoir les retrouver de manière rapide et efficace. Pour des traitements plus génériques, plusieurs langages sont utilisables, et nous pouvons citer en particulier Python [10], Java [11] ou encore Scala [12]. Ces langages peuvent être utilisés dans un calepin électronique (*notebook*), comme Jupyter [13] par exemple. Nous trouvons aussi des ETL « classiques », comme Talend [14], qui sont généralement plutôt utilisés pour effectuer des analyses BI, notamment alimenter un entrepôt de données. Enfin, des technologies plus complètes

5. `https://redis.io/`
6. `https://www.mongodb.com/`
7. `https://hypertable.org/`
8. `https://neo4j.com/`
9. `https://www.elastic.co/elasticsearch/`
10. `https://www.python.org/`
11. `https://www.java.com/`
12. `https://www.scala-lang.org/`
13. `https://jupyter.org/`
14. `https://www.talend.com/`

sont disponibles, comme Databricks [15], qui permettent à la fois de stocker mais aussi de retravailler les données.

Il existe aussi des outils permettant de faciliter le déploiement d'une infrastructure de lac de données. Par exemple, pour créer du lien entre différentes technologies introduites dans cette section, il est possible de développer une API qui permet de connecter les services entre eux. Pour cela, un langage de programmation web, comme JavaScript [16], est une bonne solution. Ce même langage permet aussi de développer des interfaces web, pour justement interagir avec l'API. Pour déployer le lac de données, il est possible d'utiliser un gestionnaire de conteneurs tel que Docker [17]. Ces conteneurs peuvent ensuite être contrôlés de manière automatisée par un outil d'orchestration comme Kubernetes [18]. Finalement, des outils tels que Grafana [19] donnent la possibilité de contrôler et surveiller tous les systèmes qui composent l'infrastructure du lac.

2.5.2.2 Ecosystème Apache

La fondation Apache [20] est une organisation à but non lucratif qui vise à proposer des logiciels *open source*, sous la licence Apache [Liu et al., 2021, Sawadogo and Darmont, 2021, John and Misra, 2017]. Les outils et technologies proposés constituent un véritable écosystème, et il est possible de mettre en place un lac de données dans son intégralité avec ces solutions. Historiquement, la solution la plus connue de la fondation est Apache HTTP server [21], qui a été pendant longtemps le serveur HTTP le plus utilisé.

Entreposage et exploitation des données Plus récemment, la fondation Apache propose tout un écosystème de logiciels et de frameworks pour l'entreposage et le traitement de mégadonnées. En particulier, le framework Hadoop [22] est une référence pour l'entreposage de données extrêmement volumineuses. Hadoop repose sur deux éléments principaux : d'une part, un système de fichiers distribué, nommé *Hadoop Distributed File System* (HDFS), qui découpe les données en blocs puis les dispatche dans les différents nœuds du *cluster*. D'autre part, MapReduce est le système de traitement qui permet de récupérer les données distribuées de manière efficace en distribuant aussi les calculs dans chaque nœud. La gestion des nœuds du *cluster* est assurée par le gestionnaire de ressources nommé YARN (*Yet Another Resource Negotiator*).

Autour de Hadoop, de nombreux autres outils sont disponibles. Pour ingérer les données, Flink [23] et Samza [24] permettent de gérer les données de flux de manière distribuée,

15. https://databricks.com/
16. https://www.javascript.com/
17. https://www.docker.com/
18. https://kubernetes.io/
19. https://grafana.com/
20. https://www.apache.org/
21. https://httpd.apache.org/
22. https://hadoop.apache.org/
23. https://flink.apache.org/
24. https://samza.apache.org/

tandis que Sqoop [25] est tout indiqué pour capter les données par lots. Flume [26] permet quant à lui de récupérer de grandes quantités de *logs*. Sqoop et Flume peuvent être utilisés efficacement en duo, le premier gérant majoritairement les données structurées et des besoins ponctuels d'ingestion, tandis que le second est plus utile pour les données semi-structurées et non structurées, générées plutôt de manière périodique. Si certaines tâches d'ingestion de données sont périodiques, elles peuvent être automatisées avec des outils comme NiFi [27] ou Airflow [28].

Nous avons évoqué que le système de stockage des données est basé sur HDFS dans l'écosystème Hadoop. Toutefois, il existe des outils qui se placent au dessus de HDFS, comme une surcouche, pour simplifier l'entreposage et l'interrogation des données structurées et semi-structurées. Parmi ces outils, nous trouvons par exemple Hive [29] qui permet de stocker des données structurées sous forme de tables, comme dans une base de données relationnelles classiques, mais aussi HBase [30] et Cassandra [31] qui sont des bases de données orientées colonnes, ce qui facilite les agrégations de données. Il existe aussi CouchDB [32], une base de données orientée documents qui est particulièrement utilisée pour sauvegarder efficacement les données semi-structurées.

Pour nettoyer et préparer les données en vue de répondre aux besoins d'analyse, le framework le plus utilisé est Spark [33], un moteur d'exécution pour effectuer des tâches de traitement de données à grande échelle. Spark permet d'écrire des programmes en Scala, Java et Python, bien que Scala soit le plus efficace pour Spark. Il est aussi possible d'utiliser le *notebook* web Zeppelin [34], qui se connecte à Spark pour visualiser de manière plus agréable les données qui sont traitées par le(s) script(s). Zeppelin permet aussi d'utiliser d'autres langages, comme SQL par exemple. Dans un registre différent, Solr [35] est un moteur de recherche basé sur l'indexation des données, qui permet d'effectuer des recherches rapides au sein des données.

Gestion et gouvernance La suite Apache propose aussi des outils pour gérer la gouvernance des données. Atlas [36] est un *framework* de gouvernance des données et de gestion des métadonnées. Grâce aux métadonnées, il est possible de construire un catalogue de données qui permet de rechercher et de filtrer les données par le biais de différents attributs de métadonnées, d'organiser les données à l'aide de classifications définies et de retracer le lignage des données. Atlas peut donc s'avérer être une pièce maîtresse lors de l'implémentation du système de métadonnées d'un lac de données.

25. `https://sqoop.apache.org/`
26. `https://flume.apache.org/`
27. `https://nifi.apache.org/`
28. `https://airflow.apache.org/`
29. `https://hive.apache.org/`
30. `https://hbase.apache.org/`
31. `https://cassandra.apache.org/`
32. `https://couchdb.apache.org/`
33. `https://spark.apache.org/`
34. `https://zeppelin.apache.org/`
35. `https://solr.apache.org/`
36. `https://atlas.apache.org/`

Il existe aussi des outils pour sécuriser l'accès au lac de données. Le rôle de cette couche de sécurité est de garantir l'authenticité de l'utilisateur, la confidentialité et l'intégrité des données. Un outil comme Ranger [37] permet d'implémenter une telle couche de gestion de la sécurité au sein du lac. La sécurité des données et la gestion des droits des utilisateurs s'inscrit aussi dans la gouvernance des données.

Nous avons introduit ici de nombreux outils et services au sein de l'écosystème Apache. Pour que ceux-ci fonctionnent en harmonie, il est nécessaire d'avoir des outils de gestion. Pour cela, Kafka [38] est une plateforme de diffusion d'évènements. Des applications peuvent produire des évènements, ou messages, qui sont enregistrés dans une file d'évènements appelée topic. D'autres applications peuvent ensuite consommer ces évènements dans les différents topics. À son arrivée sur le marché, Kafka était exclusivement un système de messagerie entre applications, mais depuis Kafka a évolué et peut désormais servir à ingérer des données de flux ou par lots ainsi qu'effectuer des traitements sur les données ingérées. Enfin, nous pouvons aussi citer Ambari [39] qui fournit une interface web de gestion d'un *cluster* Hadoop pour surveiller et gérer les différents serveurs d'une manière plus simple et agréable qu'à travers des lignes de commandes.

Distributions basées sur Hadoop Implémenter un lac de données à l'aide de technologies de l'écosystème Apache peut s'avérer être une tâche complexe à mener à bien. En effet, de nombreux outils et *frameworks* sont disponibles, certains proposant des fonctionnalités très proches, ce qui peut parfois créer de la confusion. En outre, faire communiquer tous les services, et s'assurer de leur bonne santé à travers des mises à jour régulières, tout en veillant sur le bon état des serveurs physiques, est un travail de maintenance chronophage et complexe.

C'est pourquoi des entreprises ont proposé leur propre distribution basée sur Hadoop : nous pouvons citer, entre autres, Cloudera [40] et Hortonworks. Bien que les deux compagnies aient aujourd'hui fusionné en une seule entité regroupée au sein de Cloudera, elles proposaient initialement des solutions distinctes. Cloudera proposait *Cloudera Distribution for Hadoop* [41] (CDH), et Hortoworks proposait *Hortonworks Data Platform* [42] (HDP). Depuis 2019 et la fusion des deux entreprises, la distribution proposée est *Cloudera Data Platform* [43] (CDP).

2.5.2.3 *Clouds* propriétaires

En marge de l'écosystème Apache basé essentiellement sur Hadoop, et composé de technologies ouvertes, des grands groupes ont aussi proposé leur solution pour l'entrepo-

37. https://ranger.apache.org/
38. https://kafka.apache.org/
39. https://ambari.apache.org/
40. https://www.cloudera.com/
41. https://www.cloudera.com/products/open-source/apache-hadoop/key-cdh-components.html
42. https://fr.cloudera.com/products/hdp.html
43. https://www.cloudera.com/products/cloudera-data-platform.html

sage et l'exploitation de mégadonnées. Les deux différences principales sont les suivantes : premièrement, ce sont des services proposés dans le *cloud* (ou « nuage ») contrairement à un écosystème Apache qui doit être installé sur des serveurs sur place (*on-premise*). Deuxièmement, les *clouds* propriétaires sont des services payants et fermées, tandis que les technologies Apache sont ouvertes et gratuites. Nous présentons maintenant les deux *clouds* principaux disponibles sur le marché [Kamal et al., 2020, Dutta and Dutta, 2019, Valnaos, 2019, Maurya et al., 2021, Sirosh, 2016].

Amazon Web Services Amazon propose son *cloud* baptisé *Amazon Web Services*[44] (AWS). Leader sur le marché des *clouds* propriétaires, AWS se démarque par la multitude de services mis à disposition, en constante évolution, et son large éventail de possibilités. De nombreux cas d'usages peuvent être traités dans un écosystème AWS, ce qui en fait une excellente solution pour l'implémentation d'un lac de données.

Le stockage des données dans AWS se base essentiellement sur *Simple Storage Service*[45] (S3), qui peut enregistrer tous types de données. S3 constitue généralement la pièce principale pour implémenter la zone de stockage d'un lac de données. Nous pouvons aussi citer *Elastic File System*[46] (EFS) et *Elastic Block Store*[47] (EBS), qui sont des services spécifiquement dédiés pour enregistrer des fichiers et des données en blocs, respectivement. Enfin, S3 Glacier[48] est une option pour l'enregistrement des données d'archives, c'est-à-dire des données qui sont peu ou pas utilisées.

Pour ingérer les données, AWS propose Glue[49] qui se définit comme un ETL, bien qu'il soit aussi utile pour simplement transférer des données sans les modifier. En surcouche d'un stockage des données dans S3, AWS propose des services pour exploiter les données de manière plus efficace. Par exemple, Athena[50] permet d'effectuer des requêtes SQL sur les données structurées enregistrées dans S3. De plus, Redshift[51] permet de mettre en place un entrepôt de données. Nous pouvons citer aussi Aurora[52] qui est une base de données relationnelle, ainsi que DynamoDB[53], une base de données « clés-valeurs ». Pour traiter les données avec des transformations ponctuelles mais qui peuvent être périodiques, Lambda permet d'exécuter des scripts écrits dans des langages supportés (comme Java, JavaScript ou encore Python). Pour des besoins plus conséquents qui nécessitent une machine dédiée, *Elastic Compute Cloud*[54] (EC2) propose des instances de machines virtuelles dans lesquelles il est possible de faire tourner de nombreuses applications.

44. `https://aws.amazon.com/`
45. `https://aws.amazon.com/s3/`
46. `https://aws.amazon.com/efs/`
47. `https://aws.amazon.com/ebs/`
48. `https://aws.amazon.com/s3/glacier/`
49. `https://aws.amazon.com/glue/`
50. `https://aws.amazon.com/athena/`
51. `https://aws.amazon.com/redshift/`
52. `https://aws.amazon.com/rds/aurora/`
53. `https://aws.amazon.com/dynamodb/`
54. `https://aws.amazon.com/ec2/`

Nous avons introduit Glue, l'ETL d'AWS qui permet d'ingérer les données. Ce service dispose d'une autre composante importante, s'appelant Glue Data Catalog, qui permet la mise en place d'un catalogue de métadonnées. Il permet de retracer quelles données sont transférées par qui et comment, tout en pouvant ajouter d'autres informations comme des descriptions, ce qui permet *in fine* de créer une gouvernance des données. Pour sécuriser l'accès aux différents services, ainsi qu'aux données et leurs traitements, le service *Identity and Access Management*[55] (IAM) oblige les utilisateurs à s'identifier et leur alloue ensuite des autorisations selon leurs droits prédéfinis. Les différents services AWS sont amenés à communiquer entre eux grâce à *Simple Queue Service*[56] (SQS). Enfin, le Marketplace[57] permet de gérer les services et notamment en ajouter des nouveaux.

AWS est néanmoins critiquable sur certains aspects. Premièrement, il est majoritairement orienté sur le *cloud* public, ce qui peut être un problème pour des entreprises souhaitant mettre un place un *cloud* privé, ou hybride. Aussi, la tarification des différents services est parfois difficile à comprendre, ce qui peut être problématique pour estimer un budget pour la mise en place d'une architecture répondant à un nouveau besoin.

Microsoft Azure Microsoft dispose aussi de son *cloud* nommé *Azure*[58]. Il est en concurrence directe avec AWS et est le deuxième principal *cloud* disponible sur le marché. Azure propose lui aussi de nombreux services pour enregistrer et analyser les données. La force principale d'Azure est que de nombreuses entreprises utilisent déjà des éléments de l'univers Microsoft, comme Windows, Office ou encore SQL server, et Azure se marie idéalement avec ceux-ci. De plus, Azure se destine un peu plus aux entreprises qu'AWS, et est plutôt orienté *cloud* hybride. Azure se trouve donc être une solution viable pour implémenter convenablement un lac de données. Le stockage des données dans un écosystème Azure passe essentiellement par Blob Storage[59], qui est l'équivalent d'S3 chez Amazon. Il est aussi possible d'enregistrer de manière plus efficace des fichiers avec Files[60].

Pour transférer des données, mais aussi les transformer et les préparer si besoin, le service Data Factory[61], qui est en réalité un ETL, est tout indiqué. Aussi, SQL database[62] permet de créer une base de données relationnelle qui se superpose à Blob Storage, tandis que CosmosDB[63] est une base de données de type « clés-valeurs » bien adaptée pour les données semi-structurées. Pour créer un entrepôt de données dans Azure, Synapse Analytics[64] est tout à fait adapté, bien qu'il propose aussi d'autres fonctionnalités. De plus, comme dans AWS, le service Functions[65] propose de lancer des traitements ponctuels,

55. https://aws.amazon.com/iam/
56. https://aws.amazon.com/sqs/
57. https://aws.amazon.com/marketplace/
58. https://azure.microsoft.com/
59. https://azure.microsoft.com/services/storage/blobs/
60. https://azure.microsoft.com/services/storage/files/
61. https://azure.microsoft.com/services/data-factory/
62. https://azure.microsoft.com/services/sql-database/
63. https://azure.microsoft.com/services/cosmos-db/
64. https://azure.microsoft.com/services/synapse-analytics/
65. https://azure.microsoft.com/services/functions

tandis que des machines virtuelles (*Virtual Machines*[66]) sont disponibles si un besoin d'analyse nécessite plus de ressources.

Pour mener une gouvernance des données qui sont enregistrées dans les différents systèmes mentionnés plus hauts, Azure propose Purview[67] ainsi que Data Catalog[68], qui permettent de tenir un catalogue de métadonnées, et retracer le cycle de vie des données. Concernant la gestion des accès aux services et la sécurisation des données, Azure, tout comme AWS, propose son propre service Active Directory[69] (AD). De plus, Service Bus[70] permet aux différents services de communiquer entre eux, c'est-à-dire produire et consommer des messages. Enfin, comme pour AWS, un Marketplace[71] est disponible pour gérer et administrer les différents services mis en place.

Azure présente toutefois des faiblesses, en particulier sur les ressources d'aide à l'utilisation. L'assistance technique et les formations disponibles ne sont pas systématiquement au niveau, et la documentation existante peut parfois manquer de contenu, plus particulièrement pour les services récents.

Des éditeurs ont aussi proposé des surcouches qui viennent exploiter des services proposés dans le cloud. Par exemple, le *Delta Lake*, proposé par Databricks[72], est une surcouche qui se place au dessus des objets de stockage cloud, comme Azure Blob Storage ou Amazon S3 [Armbrust et al., 2020]. Cette surcouche permet de créer des *Delta tables*, qui gèrent les propriétés ACID (atomicité, cohérence, isolation et durabilité) pour garantir que les transactions exécutées soient fiables. Les données sont enregistrées au format Parquet, et contiennent aussi les autres informations nécessaires au bon fonctionnement des tables Delta. Grâce à ce fonctionnement, le *Delta lake* permet aux utilisateurs d'exploiter les données de manière très efficiente, mais aussi dans certains cas de simplifier une architecture existante, qui peut présenter parfois de la redondance.

En définitive, de nombreux outils existent pour implémenter un lac de données, qu'ils soient issus de l'écosystème Apache, des *clouds* propriétaires, ou encore des outils autonomes. Il est intéressant de noter que les technologies Apache peuvent être utilisées de manière autonome, et donc être associées à des technologies indépendantes par exemple. Notons toutefois que les outils Apache sont avant tout étudiés pour fonctionner avec d'autres outils Apache, et qu'il faudrait parfois un certain travail pour les faire communiquer avec d'autres outils qui ne sont initialement pas prévus pour. De même, certains services proposés par les *clouds* propriétaires peuvent eux aussi être utilisés de manière autonome et donc intégrer un assemblage de technologies ouvertes. Mais à nouveau, ces services sont faits pour travailler ensemble, et selon les cas, associer des services cloud à des technologies ouvertes peut nécessiter un travail supplémentaire lors de l'implémentation.

66. `https://azure.microsoft.com/services/virtual-machines/`
67. `https://azure.microsoft.com/services/purview/`
68. `https://azure.microsoft.com/services/data-catalog/`
69. `https://azure.microsoft.com/services/active-directory/`
70. `https://azure.microsoft.com/services/service-bus/`
71. `https://azure.microsoft.com/marketplace/`
72. `https://databricks.com/`

2.6 Conclusion

Dans cet état de l'art, nous avons étudié en détail le concept de lac de données, une approche nouvelle pour le stockage et l'exploitation de données hétérogènes et massives. Alors que cette appellation est encore récente puisque sa première mention remonte à 2010, il n'y a pas encore de véritable consensus scientifique sur la définition concrète d'un lac de données. C'est pourquoi nous avons proposé notre définition du concept de lac de données, et cette définition nous permet ensuite de la comparer à d'autres systèmes d'information, comme les entrepôts de données.

Nous devons préciser que l'ordre des sections de ce chapitre ne suit pas l'ordre des contributions qui constituent la suite du manuscrit. En effet, nous avons commencé par étudier les métadonnées du lac au sens large, et notamment comment les constituer selon le type des données. Nous nous sommes ensuite penchés sur la question de l'organisation de ces métadonnées dans un système de métadonnées. Enfin, nous avons passé en revue les propositions d'architecture ainsi que les technologies susceptibles d'implémenter un lac de données. En revanche, l'ordre de nos contributions est le suivant : nous commençons par présenter notre modélisation des métadonnées, qui est une proposition assez théorique. Nous introduisons ensuite notre architecture et notre implémentation de lac de données, qui prend justement appui sur notre modèle de métadonnées. Enfin, nous détaillons un module spécifique de notre implémentation qui sert à décrire de manière plus poussée les données structurées à leur insertion dans le lac. Nous avons choisi cet ordre pour suivre une logique allant de la proposition la plus générale à la plus spécifique.

Le premier problème abordé dans cet état de l'art est celui du contenu des métadonnées, et de comment elles permettent de décrire efficacement les données. Nous distinguons deux types de métadonnées. D'une part, nous avons les métadonnées universelles, qui peuvent décrire toutes les données du lac. Elles prennent en général la forme de propriétés ou d'informations sémantiques qui se retrouvent dans toutes les données. D'autre part, nous avons les métadonnées spécifiques en fonction du type des données. En effet, décrire de manière détaillée le contenu d'un fichier CSV ne se fera pas de la même façon que décrire le contenu d'un fichier audio. Chaque type de données possède donc ses descripteurs spécifiques.

Néanmoins, il n'est pas garanti que la création des métadonnées soit une tâche rapide et facile pour l'utilisateur. De plus, constituer des descripteurs en fonction du type des données n'est pas une tâche triviale, qui plus est pour un utilisateur qui n'est pas un expert des données qu'il manipule. C'est pourquoi nous proposons un assistant à la création de métadonnées spécifiquement dédiées aux données structurées générées de manière périodique, mais dont le schéma est susceptible d'évoluer lors d'une nouvelle occurrence des données. Cette contribution, répondant à un besoin métier soulevé dans le contexte de l'entreprise, est présentée dans le chapitre 5.

Parmi les problèmes de recherche autour du lac de données, celui de l'organisation des métadonnées est particulièrement d'actualité. Une bonne organisation des métadonnées

est essentielle au bon fonctionnement d'un lac de données, pour permettre aux utilisateurs de retrouver facilement les données qu'il contient. Plusieurs systèmes de métadonnées ont été proposés dans la littérature scientifique, certains étant des implémentations de lacs de données, peu détaillées et donc difficilement réutilisables. Les autres propositions sont des modèles de métadonnées, qui se veulent plus génériques et *a priori* réutilisables sur des cas d'usage différents.

Toutefois, aucun système de métadonnées, implémentation ou modèle, ne prend en charge les six fonctionnalités clés que nous avons identifié pour les comparer. Ceci nous a motivé à proposer un nouveau modèle de métadonnées, nommé MEDAL, qui propose l'ensemble des 6 fonctionnalités. Des travaux plus récents ont vu émerger de nouveaux modèles de métadonnées et nous ont poussé à revoir notre modèle pour proposer goldMEDAL, notre second modèle de métadonnées, qui se veut plus générique que les propositions actuelles de la littérature. Ces contributions concernant l'organisation des métadonnées dans un lac de données font l'objet du chapitre 3.

Enfin, la question des architectures de lacs de données a été abordée dans cet état de l'art. En marge de l'organisation des métadonnées, des équipes de recherche ont en effet étudié la problématique des composants principaux qu'un lac de données peut proposer. Trois architectures principales émergent : les bassins de données qui séparent les données selon leur type, les zones pour compartimenter les données en fonction de leur maturité, et les objets qui rassemblent les données en fonction de leur signification sémantique. En revanche, ces architectures ont selon nous trop tendance à compartimenter les données du lac, ce qui est contraire au concept de lac de données tel que nous le voyons. C'est pourquoi nous proposons notre architecture de lac de données dans le chapitre 4.

En outre, nous avons également présenté divers outils et technologies qui permettent d'implémenter de telles architectures. Des écosystèmes complets existent, comme Apache dont le code source est libre d'accès, ou les *clouds* propriétaires proposés notamment par Amazon et Microsoft. Il est aussi possible d'assembler des technologies individuels, par exemple des bases de données NoSQL, pour constituer une implémentation ad-hoc. À nouveau, nous apportons une contribution dans le chapitre 4 avec une nouvelle implémentation de lac de données.

Chapitre 3

MEDAL et goldMEDAL, modélisation des métadonnées pour lac de données

Sommaire

"When Kobe Bryant died, a piece of me died. And as I look in this arena and across the globe, a piece of you died, or else you wouldn't be here. Those are the memories that we have to live with and we learn from."

Michael Jordan (2020)

Résumé

Un lac de données efficace nécessite un système de métadonnées qui répond aux nombreux problèmes posés par le traitement des mégadonnées. En conséquence, l'étude des systèmes de métadonnées des lacs de données est actuellement un sujet de recherche actif et de nombreuses propositions ont été faites à cet égard. Toutefois, les systèmes de métadonnées existants sont soit des implémentations de lac de données adaptées à un cas d'usage spécifique, soit des modèles de métadonnées, explicites mais insuffisamment génériques pour gérer différents types de lacs de données.

Dans ce chapitre, nous proposons une typologie de métadonnées, reposant sur la notion d'objet qui désigne un ensemble homogène de données, et qui rassemble les métadonnées en trois catégories (intra, inter, globales). Cette typologie nous permet ensuite de proposer MEDAL, notre premier modèle de métadonnées, qui adopte une modélisation logique à base de graphes. MEDAL gère les six fonctionnalités des systèmes de métadonnées, mais en le confrontant aux modèles de métadonnées les plus récents de l'état de l'art, nous constatons que MEDAL, comme les autres modèles, manque de généricité sur certains aspects.

Pour pallier ce problème, nous faisons évoluer MEDAL pour proposer un métamodèle de métadonnées baptisé goldMEDAL. Ce nouveau métamodèle adopte une modélisation en trois niveaux : conceptuel, logique et physique. À nouveau, nous comparons goldMEDAL aux modèles de métadonnées les plus récents et montrons que nous pouvons reproduire ces modèles de métadonnées avec les concepts de goldMEDAL. En outre, nous démontrons comment goldMEDAL englobe toutes les caractéristiques de modélisation des métadonnées.

3.1 Introduction

Avec la montée en popularité des lacs de données, plusieurs équipes ont commencé à s'attaquer aux problèmes de recherche inhérents à ceux-ci [Madera and Laurent, 2016, Miloslavskaya and Tolstoy, 2016]. L'un des principaux est la gestion efficace des métadonnées pour éviter que les lacs de données ne se transforment en marécages de données inexploitables [Inmon, 2016, Suriarachchi and Plale, 2016, Khine and Wang, 2017, Quix and Hai, 2018, Sawadogo and Darmont, 2021].

Cependant, la plupart des propositions de gestion des métadonnées dans la littérature [Hai et al., 2016, Beheshti et al., 2018, Mehmood et al., 2019], ainsi que les implémentations associées, fournissent peu de détails sur la façon dont les métadonnées sont organisées, rendant ces travaux difficilement réutilisables. C'est pourquoi des modèles de métadonnées ont vu le jour, visant à fournir des directives détaillées pour la conception de systèmes de métadonnées tout en étant génériques, c'est-à-dire flexibles et adaptables à de nombreux cas d'usage. Pourtant, la modélisation générique des métadonnées de lac de données reste un sujet de recherche ouvert. Une évaluation (menée dans le chapitre 2) basée sur les fonctionnalités des systèmes de métadonnées montre en effet qu'aucun d'entre eux ne les propose toutes, modèles y compris.

Par conséquent, nous commençons ce chapitre en proposant une typologie de métadonnées, qui nous permet d'identifier les métadonnées qui constituent le système de métadonnées du lac. Elle s'appuie sur la notion d'objet pour désigner un ensemble homogène de données, et divise les métadonnées en trois catégories : intra-objet, inter-objets et globales. Nous utilisons ensuite cette typologie pour proposer un premier modèle de métadonnées, nommé *MEtadata model for DAta Lakes* (MEDAL). Ce modèle adopte une représentation logique à base de graphes, et les objets sont notamment modélisés par des hypernœuds. De plus, un modèle physique de MEDAL a été implémenté dans le cadre d'un projet mené par d'autres membres du laboratoire ERIC. Toutefois, une comparaison avec les modèles de métadonnées plus récents que MEDAL nous pousse à revoir nos critères d'évaluation des modèles de métadonnées [Eichler et al., 2020]. À la lumière de ces nouveaux éléments, nous avons réalisé que MEDAL n'était pas suffisamment générique pour gérer toutes les problématiques de modélisation des métadonnées. En réponse, nous proposons le métamodèle goldMEDAL, une évolution de MEDAL définie par un processus de modélisation classique à trois niveaux, c'est-à-dire conceptuel, logique et physique. Nous choisissons une représentation formelle conceptuelle pour éviter toute ambiguïté, mais fournissons également une représentation UML pour la lisibilité. Le niveau logique est une traduction des concepts à l'aide de la théorie des graphes. En outre, nous décrivons deux modèles physiques différents comme preuve de concept. Finalement, nous montrons que les concepts de goldMEDAL généralisent presque tous ceux des modèles de métadonnées de l'état de l'art, et couvrent toutes les caractéristiques que nous avons utilisées pour comparer les modèles de métadonnées des lacs de données.

Notons que ces propositions, à savoir la typologie de métadonnées, le modèle MEDAL et le métamodèle goldMEDAL, sont le fruit d'un travail mené en collaboration avec les « *datalakers* », une petite équipe de chercheurs du laboratoire ERIC incluant en

particulier Pegdwendé N. Sawadogo, qui a travaillé sur ce sujet dans le cadre de sa thèse de doctorat. Trois projets (dont le nôtre) ont été menés au laboratoire ERIC, dans lesquels le lac de données est une problématique clé. C'est en échangeant en équipe sur ces projets, chacun ayant leurs motivations, leurs objectifs et leurs contraintes, que nous avons pu viser à plus de généricité, en particulier pour proposer le métamodèle de métadonnées goldMEDAL. Notons qu'en ce qui concerne notre cas d'usage, il est présenté et discuté dans les chapitres 4 et 5.

La présentation de MEDAL a fait l'objet de deux publications : P.N. Sawadogo, E. Scholly, C. Favre, E. Ferey, S. Loudcher, J. Darmont, "Metadata Systems for Data Lakes : Models and Features", *1st International Workshop on BI and Big Data Applications (BBIGAP@ADBIS 2019), Bled, Slovenia*, September 2019 ; *Communications in Computer and Information Science*, Vol. 1064, Springer, Heidelberg, Germany, 440-451 [Sawadogo et al., 2019b], et E. Scholly, P. N. Sawadogo, C. Favre, E. Ferey, S. Loudcher, J. Darmont, "Systèmes de métadonnées dans les lacs de données : modélisation et fonctionnalités", *15e journées EDA Business Intelligence & Big Data (EDA 2019), Montpellier*, Octobre 2019 ; *Revue des Nouvelles Technologies de l'Information*, Vol. B-15, 77-92 [Scholly et al., 2019]. En outre, la présentation de goldMEDAL a aussi été publiée dans un article. E. Scholly, P.N. Sawadogo, P. Liu, JA Espinosa-Oviedo, C. Favre, S. Loudcher, J. Darmont, C. Noûs, "Coining goldMEDAL : A New Contribution to Data Lake Generic Metadata Modeling", *23rd International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP@EDBT/ICDT 2021), Nicosia, Cyprus*, March 2021 ; *CEUR*, Vol. 2840, 31-40 [Scholly et al., 2021b].

Le reste de ce chapitre est organisé comme suit. Nous commençons par présenter dans la section 3.2 une typologie de métadonnées, qui les partage en trois catégories : intra, inter et globales. En prenant appui sur cette typologie, nous présentons dans la section 3.3 notre premier modèle de métadonnées MEDAL. Ensuite, la section 3.4 décrit comment MEDAL permet de gérer les six fonctionnalités identifiées pour les systèmes de métadonnées, mais montre aussi ses limitations en le comparant à des modèles de métadonnées plus récents. En réponse à ces limites, la section 3.5 présente le métamodèle goldMEDAL, notre seconde proposition pour la modélisation des métadonnées, se voulant plus générique. Nous y détaillons ses métamodèles conceptuel et logique, ainsi que deux modèles physique. Nous illustrons dans la section 3.6 comment goldMEDAL répond aux caractéristiques de modélisation des métadonnées et généralise les modèles de métadonnées des lacs de données existants, et nous détaillons également comment goldMEDAL peut être appliqué à quelques cas d'utilisation réels. Finalement, la section 3.7 conclut ce chapitre.

3.2 Typologie de métadonnées

Rappelons la définition de métadonnées définie par [Riley, 2017] : « Les métadonnées sont des informations structurées qui décrivent, expliquent, localisent ou encore facilitent ou permettent la recherche, l'utilisation ou la gestion d'une ressource d'information ». Dans l'optique de cadrer notre travail de modélisation des métadonnées dans le contexte

des lacs, nous nous sommes penchés sur la question des informations servant à constituer les métadonnées, afin qu'elles soient les plus complètes possible. C'est pourquoi nous proposons dans cette section une typologie de métadonnées.

Pour l'introduire, il est primordial de définir un concept générique représentant tout ensemble homogène de données que le modèle doit traiter. Certains travaux sur les lacs de données ont proposé les concepts d'unité de données [Quix et al., 2016], d'entité [Beheshti et al., 2017], de jeu de données (*dataset*) [Maccioni and Torlone, 2018] et d'objet [Diamantini et al., 2018]. Nous adoptons la notion d'objet, qui nous semble plus appropriée pour représenter de façon abstraite un ensemble de données. Plus concrètement, un objet peut se matérialiser par une table relationnelle ou un fichier physique (document de tableur, XML ou JSON, document textuel, collection de tweets, image, vidéo, etc.).

Le reste de cette section présente notre typologie des métadonnées d'un lac de données, qui est une extension de celle proposée par [Sawadogo et al., 2019a] que nous complétons en prenant en compte de nouveaux types de métadonnées. Notre typologie organise les métadonnées en trois catégories : intra, inter et globales. La section 3.2.1 présente les métadonnées intra, c'est-à-dire les métadonnées contenues au sein d'un objet. Après cela sont introduites les métadonnées inter dans la section 3.2.2, i.e. les métadonnées permettant d'interconnecter les objets. La présentation de cette typologie se termine dans la section 3.2.3 avec les métadonnées globales.

3.2.1 Métadonnées intra

Cette catégorie désigne les métadonnées associées à un objet précis. Nous en distinguons plusieurs types.

Les **propriétés** fournissent une description générale de l'objet, sous la forme de couples clé-valeur. Ces métadonnées sont généralement obtenues à partir du système de fichiers : titre de l'objet, taille, date de dernière modification, chemin d'accès, etc.

Les **résumés et prévisualisations** ont pour rôle de donner un aperçu du contenu ou de la structure d'un objet. Elles peuvent prendre la forme d'un schéma des données dans un contexte de données structurées ou semi-structurées, ou d'un nuage de mots pour des données textuelles.

Les données brutes dans le lac sont souvent amenées à être modifiées à travers des mises à jour. De telles opérations entraînent la création de nouvelles **versions** des données initiales, qui peuvent être considérées comme des métadonnées. De même, les données brutes (surtout non structurées) peuvent être reformatées pour un usage spécifique. Cette opération induit la création d'une nouvelle **représentation** de l'objet.

Les **métadonnées sémantiques** sont des annotations qui permettent de comprendre le sens des données. Il s'agit plus concrètement de *tags* descriptifs, de descriptions textuelles ou de catégorisations métier. Les métadonnées sémantiques servent souvent de base à la détection de relations entre les données du lac.

3.2.2 Métadonnées inter

Les métadonnées inter-objets traduisent les liaisons entre les objets. Chacune de ces métadonnées est donc associée à au moins deux objets. Nous en distinguons trois types : les regroupements d'objets, les liaisons de similarité et les relations de parenté.

Les **regroupements d'objets** consistent à organiser les objets du lac en collections, chaque objet pouvant appartenir simultanément à plusieurs collections. Ces regroupements peuvent être déduits automatiquement des métadonnées sémantiques telles que des *tags* et des catégories métier. Certaines propriétés peuvent également servir de base à la génération des regroupements ; les objets peuvent ainsi être rassemblés suivant leur format ou leur langue d'édition, par exemple.

Les **liaisons de similarité** traduisent la force de la ressemblance entre deux objets. À l'inverse des regroupements d'objets, les relations de similarité portent sur les propriétés intrinsèques des objets, notamment leur contenu ou leur structure. Par exemple, il peut s'agir du taux de mots en commun entre deux documents textuels, d'une mesure de la compatibilité des schémas de deux objets structurés ou semi-structurés [Maccioni and Torlone, 2018], ou d'autres mesures de similarité usuelles.

Les **relations de parenté**, que nous ajoutons à la typologie de [Sawadogo et al., 2019a], traduisent le fait qu'un objet peut être issu de la jointure de plusieurs autres. Dans un tel cas, il existe une relation de « parenté » entre les objets combinés et l'objet résultant, et une relation de « co-parenté » entre les objets fusionnés. Ce type de relation permet ainsi de tirer parti des traitements effectués dans le lac de données pour identifier des objets utilisables conjointement, en plus de conserver une traçabilité de la provenance des objets générés à l'intérieur du lac.

3.2.3 Métadonnées globales

Les métadonnées globales sont des structures de données destinées à donner une couche de contexte aux données du lac en vue de faciliter et d'optimiser leur analyse. Contrairement aux métadonnées intra et inter, les métadonnées globales concernent potentiellement l'ensemble du lac de données. En plus des ressources sémantiques identifiées par [Sawadogo et al., 2019a], nous proposons deux nouveaux types de métadonnées globales.

Les **ressources sémantiques** sont essentiellement des bases de connaissances (ontologies, taxonomies, thésaurus, dictionnaires) utilisées à la fois pour générer d'autres métadonnées et améliorer les analyses. L'utilisation d'un thésaurus permet ainsi d'étendre une requête par mots-clés en associant des synonymes des termes saisis par l'utilisateur. De même, un thésaurus peut servir lors de la génération de regroupements de données pour fusionner des collections issues de *tags* différents mais équivalents.

Les ressources sémantiques sont généralement issues de sources externes, comme par exemple les ontologies du web des données. Notons toutefois que dans certains cas, les ressources sémantiques peuvent être constituées et personnalisées spécialement pour la gestion et l'analyse des données du lac. Une ontologie métier peut ainsi servir à définir des concepts abstraits permettant de regrouper lors de l'analyse plusieurs concepts

équivalents ou proches.

Les **index et index inversés** sont des structures de données permettant de retrouver rapidement un objet sur la base de caractéristiques précises. Ils établissent, ou mesurent, la correspondance entre ces caractéristiques (mots-clés, motifs, couleurs) et les objets contenus dans le lac de données. Les index peuvent être simples (indexation textuelle) ou plus complexes (sur le contenu d'images, de sons, etc.). Ils servent principalement à la recherche de données dans le lac.

Couramment appelés *logs*, les **journaux d'événements** permettent de tracer les interactions entre les utilisateurs et le lac de données. Cela passe par l'enregistrement séquentiel d'événements comme la connexion d'un utilisateur, la modification ou la consultation d'un objet dans un fichier ou une base de données. Ces métadonnées permettent d'analyser l'utilisation du lac de données par l'identification des objets les plus consultés ou par l'étude des comportements des utilisateurs.

3.3 Présentation du modèle MEDAL

En nous appuyant sur cette typologie ainsi que sur le concept d'objet, nous introduisons maintenant un nouveau modèle de métadonnées nommé MEDAL (*MEtadata model for DAta Lakes*).

Cette section s'organise de la manière suivante. Nous commençons par donner dans la section 3.3.1 un diagramme de classes UML qui représente de manière visuelle le modèle conceptuel de MEDAL. Par la suite, nous présentons dans la section 3.3.2 le niveau logique de MEDAL, en présentant comment se déclinent en graphe les métadonnées intra-objet, inter-objets et globales. Nous terminons par présenter un modèle physique de MEDAL dans la section 3.3.3, qui est une implémentation effectuée dans le cadre d'un cas d'usage réel.

3.3.1 Modèle conceptuel

Le modèle MEDAL se base sur la typologie de métadonnées, introduite dans la section 3.2, et reprend donc ses différentes composantes (objet, version, liaison de similarité, etc.). Sans toutefois proposer de manière explicite un modèle conceptuel, le diagramme de classe de la figure 3.1 présente de manière visuelle les concepts du modèle MEDAL par analogie aux diagrammes de classes des modèles de l'état de l'art présentés dans le chapitre 2.

Les concepts sont modélisés soit comme des classes (objet, regroupement, données, version et représentation), soit comme des classes d'association (relation de parenté, liaison de similarité, mise à jour et transformation). Notons que la classe *Données* présente sur ce diagramme n'est pas un concept explicite de MEDAL, mais elle a du sens dans ce diagramme puisqu'elle généralise les classes de version et représentation. Il est intéressant de noter que la classe objet est vraiment centrale dans ce diagramme : elle est liée à deux classes et deux classes d'association.

Les différentes classes et classes d'associations portent des attributs de tous types, mais ces derniers ne sont pas représentés sur la figure.

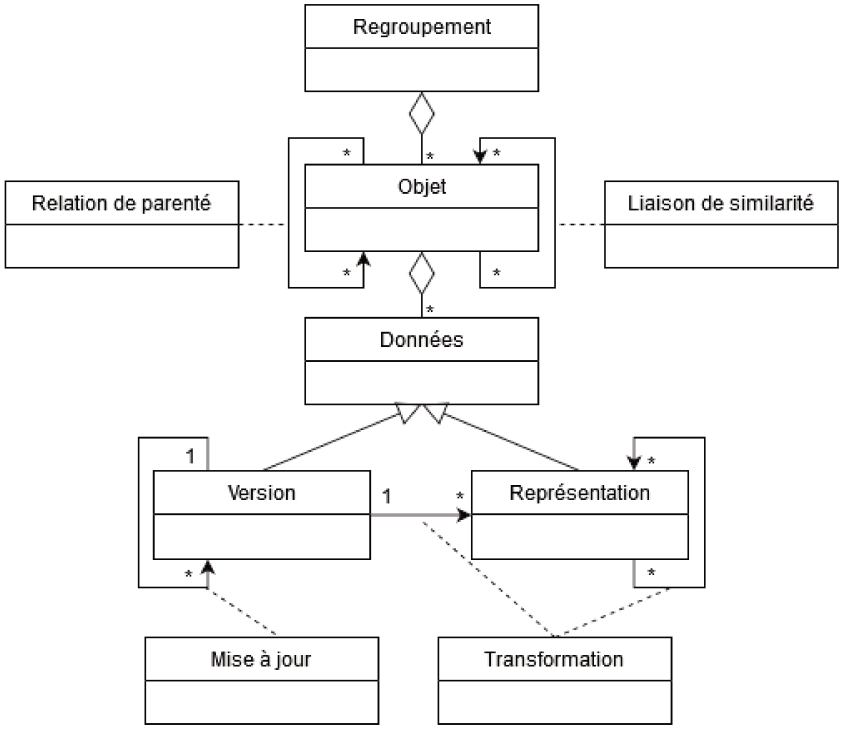

FIGURE 3.1 – Diagramme de classes UML du modèle conceptuel de MEDAL

3.3.2 Modèle logique

D'un point de vue logique, MEDAL adopte une représentation des métadonnées à base de graphes. Les graphes présentent en effet le double avantage d'offrir un schéma flexible et de faciliter l'expression de relations. Ainsi, nous représentons un objet par un **hypernœud**, qui est un nœud pouvant contenir d'autres nœuds (graphe imbriqué). Ces hypernœuds contiennent divers éléments (versions et représentations, propriétés, etc.), et peuvent être liés entre eux (similarité, parenté, etc.).

3.3.2.1 Métadonnées intra-objet

Chaque hypernœud contient une ou des **représentations**, qui traduisent le fait que les données associées à l'objet peuvent être présentées de différentes manières. Il existe *a minima* une représentation par hypernœud, qui est constitué des données brutes ingérées dans le lac de données. Les autres représentations sont toutes issues de cette représentation initiale. Chaque représentation correspond à un nœud qui possède des attributs, simples ou complexes. Ceux-ci sont les propriétés de la représentation. Notons qu'une

représentation peut être associée à un ensemble de données effectivement stocké dans le lac ou être une vue calculée à la demande.

Le passage d'une représentation à une autre se fait via une **transformation**. Elle prend la forme d'une arête orientée reliant deux nœuds de représentation. Cette arête possède aussi des attributs, qui sont les propriétés décrivant le processus de transformation ayant permis de passer de la première représentation à la seconde. Dans l'idéal, la transformation conserve le script qui a servi à cette opération ; mais si la transformation est manuelle, alors l'utilisateur ayant effectué cette transformation doit en saisir une description pour assurer une bonne compréhension de son travail par d'autres utilisateurs.

Un hypernœud peut aussi contenir des **versions**, qui servent à gérer les évolutions des données présentes dans le lac à travers le temps. Comme les représentations, nous associons les versions à des nœuds, possédant eux aussi des attributs pour y stocker leurs propriétés. La création d'un nouveau nœud de version n'est pas forcément systématique au moindre changement. Selon la nature et la fréquence d'évolution des données, il est possible de mettre en place diverses stratégies pour gérer ces évolutions. La création d'une nouvelle version se fait via une **mise à jour** semblable à une transformation, puisqu'elle est aussi traduite par une arête orientée et possède des attributs.

Enfin, comme les nœuds de représentation et de version, l'**hypernœud** est porteur d'attributs, qui permettent de le décrire. Ces attributs peuvent être des propriétés comme la provenance de l'ensemble de données, ou bien des agrégats des attributs des représentations et versions qu'il contient, par exemple le nombre de versions, de représentations, la taille cumulée, etc. Nous pouvons faire le choix d'enregistrer ces propriétés agrégées ou de les calculer à la volée.

Ainsi, un hypernœud contient un arbre dont les nœuds sont des représentations ou des versions et les arcs dirigés sont des transformations ou des mises à jour. Une représentation (resp. version) est issue d'une autre par une transformation (resp. mise à jour). Une version peut donner lieu à une représentation via une transformation, mais une version ne peut pas être issue d'une représentation. Ainsi, la racine de l'arbre est la représentation brute initiale de l'hypernœud et chaque version possède son propre sous-arbre de représentations.

Définition 2 *Soit $\mathcal{N}$ un ensemble de nœuds. L'ensemble des* métadonnées intra $\mathcal{M}_{intra}$ *est l'ensemble des hypernœuds tel que $\forall h \in \mathcal{M}_{intra}, h = \langle N, E \rangle$, où :*

- *$N \subset \mathcal{N}$ est l'ensemble des nœuds (représentations et versions) porteurs d'attributs de l'hypernœud noté h ;*
- *$E = \{r_{(transformation \mid mise\ à\ jour)} \in N \times N\}$ est l'ensemble des arcs (transformations et mises à jour) porteurs d'attributs de h.*

Nous illustrons ces notions à travers un exemple représenté par la figure 3.2. Imaginons une entreprise vendant divers produits. Les informations sur ces produits (nom, prix unitaire, description, etc.) sont stockées dans le lac sous la forme d'un fichier XML. Un hypernœud (coloré en bleu) décrit cet ensemble de données et il possède un nœud de

version v_1 (coloré en orange) qui correspond au fichier XML tel qu'initialement ingéré dans le lac. Afin d'assister l'interrogation des informations sur les produits, un utilisateur décide d'extraire le schéma du fichier XML. Cette transformation notée t_1 génère une nouvelle représentation r_1 (coloré en vert) issue de la version initiale. Supposons maintenant que les données évoluent, car le prix de certains produits a changé, et que de nouveaux produits ont été ajoutés au catalogue. Ce changement dans les données est une mise à jour notée m_1, et génère une nouvelle version v_2, liée à la première version par une mise à jour. Enfin, si l'utilisateur fait la même opération et extrait le schéma des données plus récentes, il crée alors une nouvelle représentation r_2 issue de la deuxième version v_2. L'opération reliant v_2 à r_2 est la transformation t_2.

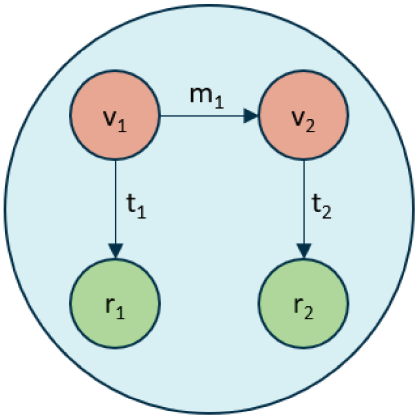

FIGURE 3.2 – Hypernœud et son arbre de versions et représentations

3.3.2.2 Métadonnées inter-objets

Un **regroupement d'objets** est modélisé par un ensemble d'hyperarcs non orientés, c'est-à-dire des arcs pouvant lier plus de deux nœuds (en l'occurrence, des hypernœuds). Chaque hyperarc correspond à une collection d'objets. Si le regroupement est effectué sur un attribut d'hypernœud, un hypernœud appartient à l'hyperarc qui correspond à sa valeur pour l'attribut. Il existe donc autant d'hyperarcs que de valeurs distinctes pour l'attribut considéré. Notons que tous les attributs ne servent pas forcément à faire des regroupements, et que des regroupements peuvent être effectués sur d'autres éléments que les attributs (par exemple, une catégorisation métier).

Une **liaison de similarité** entre deux hypernœuds est représentée par une arête non orientée porteuse d'attributs : valeur de la mesure de similarité, type de mesure utilisée, date de la mesure, etc. Pour que deux hypernœuds soient connectés par une liaison de similarité, ils doivent être comparables, c'est-à-dire qu'ils doivent contenir chacun une représentation qui peut être comparée à l'autre grâce à une mesure de similarité.

Un hypernœud peut être issu d'autres hypernœuds à travers un **lien de parenté**. Pour traduire cette relation, nous avons recours à un hyperarc orienté : l'ensemble des

hypernœuds « parents » et l'hypernœud « enfant » sont reliés par cet hyperarc orienté vers l'hypernœud enfant. Une fois encore, cet hyperarc possède des attributs descriptifs.

Définition 3 *L'ensemble des* métadonnées inter $\mathcal{M}_{inter}$ *est défini par les trois couples* $\langle H, E_g \rangle$, $\langle H', E_s \rangle$ *et* $\langle H'', E_p \rangle$ *tels que :*

- $H \subset \mathcal{M}_{intra}$, $H' \subset \mathcal{M}_{intra}$ *et* $H'' \subset \mathcal{M}_{intra}$ *sont des ensembles d'hypernœuds porteurs d'attributs ;*
- $E_g = \{E_g^{param} \mid E_g^{param} : H \rightarrow \mathcal{P}(H)\}$ *est l'ensemble des fonctions regroupant les hypernœuds dans des collections selon un paramètre donné (souvent, un attribut) ;*
- $E_s = \{s \mid s \in H' \times H'\}$ *est l'ensemble des arcs (liaisons de similarité) porteurs d'attributs ;*
- $E_p = \{(h_1, ..., h_n, h_{enfant}) \mid (h_1, ..., h_n, h_{enfant}) \in (H'')^{n+1}\}$ *est l'ensemble des liaisons de parenté, où* $(h_1, ..., h_n)$ *sont les hypernœuds parents (*$n \geq 2$*) et* h_{enfant} *l'hypernœud enfant.*

Dans l'exemple représenté graphiquement par la figure 3.3, les hypernœuds sont colorés en bleu. Nous reprenons l'hypernœud de l'exemple de la section précédente (que nous notons ici o_2), et nous ajoutons deux autres hypernœuds : le premier contient des tweets en rapport avec l'entreprise récoltés sur internet noté o_1, ainsi qu'une vidéo commerciale des produits vendus dont l'hypernœud est noté o_3. Dans un regroupement sur la provenance des données dont les collections sont colorées en orange, o_1 (tweets) est seul dans la collection « source externe », tandis que les deux autres hypernœuds o_2 et o_3 sont dans la collection « source interne ». Dans un second regroupement sur le format de la version initiale (couleur verte), c'est o_3 (vidéo) qui est seul dans la collection « non structuré », alors que les deux autres hypernœuds o_1 et o_2 sont dans la collection « semi-structuré ». Notons que les collections sont représentées par des rectangles en pointillés dans la figure, et que les attributs des hypernœuds ne sont pas représentés.

3.3.2.3 Métadonnées globales

Les métadonnées globales sont particulières, et sont donc gérées différemment des métadonnées intra-objet et inter-objets. En effet, les index et les journaux d'événements sont plutôt des structures physiques, et sont donc grandement dépendants de la technologie employée pour implémenter le lac de données et le système de métadonnées. De plus, la gestion des ressources sémantiques est une thématique déjà largement abordée dans la littérature scientifique. Pour toutes ces raisons, MEDAL ne propose pas de modélisation logique particulière pour les métadonnées globales.

3.3.3 Modèle physique

Après avoir donné le modèle logique de MEDAL, nous présentons maintenant un exemple d'implémentation de celui-ci au niveau physique. Cet exemple est tiré du cas d'usage lié à la thèse de doctorat de Pegdwendé N. Sawadogo, et au projet AURA-PMI.

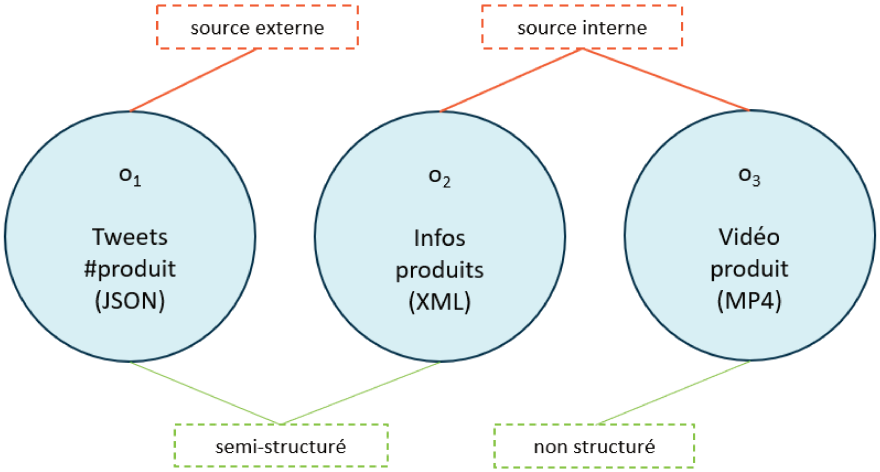

FIGURE 3.3 – Hypernœuds interconnectés

Le lac de données, appelé AUDAL, tire son nom de ce projet mené par des chercheurs en sciences de la gestion qui souhaitent analyser l'effet de la servicisation (c'est-à-dire le passage de la fourniture de produits à la fourniture de services) et de la numérisation sur les performances économiques des petites et moyennes entreprises [Sawadogo et al., 2021]. À cette fin, diverses données ont été recueillies, allant de documents textuels (rapports annuels, communiqués de presse, sites Web, messages sur les médias sociaux) à des fichiers de tableur présentant des caractéristiques qualitatives (par exemple, les stocks) et qualitatives (par exemple, le degré de servicisation). Le défi consiste alors à déduire des similarités et/ou des dissemblances entre les entreprises ciblées grâce à l'analyse des données. Une solution possible pour atteindre cet objectif est d'organiser et d'intégrer toutes les données dans un lac de données, afin que les utilisateurs, c'est-à-dire les chercheurs, puissent trouver des liens entre les entreprises à travers leurs données respectives, grâce aux métadonnées du lac.

Le système de métadonnées d'AUDAL repose sur trois bases de données NoSQL (MongoDB, Neo4j et ElasticSearch), et est organisé en trois niveaux, permettant de gérer les métadonnées intra-objet, inter-objets et globales.

Modèle physique des métadonnées intra Le premier niveau du système de métadonnées gère les objets et leurs métadonnées. Les objets, c'est-à-dire les documents textuels et documents tabulaires, peuvent posséder simultanément des données *brutes* et *raffinées*, qui sont alors modélisées par des versions et des représentations, respectivement. Les versions sont des pointeurs vers les fichiers correspondants dans leur format d'origine, mais contiennent aussi des métadonnées sous la forme d'attributs de nœuds Neo4j, comme par exemple le ou les auteur(s) du fichier, la date de création, etc. Les représentations (données raffinées) sont automatiquement générées à partir des versions (données brutes). Ces dernières sont transformées dans l'optique d'être exploitées lors d'analyses.

Plus concrètement, les données tabulaires brutes sont raffinées en tables relationnelles pour bénéficier de l'interrogation SQL. De même, les documents textuels bruts sont raffinés en sacs de mots ou en vecteurs de plongement (*embeddings*). Ces représentations de documents sont stockés dans le SGBD orienté documents MongoDB, et référencés à partir de nœuds Neo4j. La figure 3.4 est justement une capture d'écran de Neo4j : le nœud bleu en surbrillance représente un objet (ici, un document textuel), et ses attributs apparaissent au bas de la figure. Le nœud beige est un pointeur vers les données textuelles brutes, tandis que les nœuds verts pointent vers deux représentations raffinées des données, l'une sous la forme d'un sac de mots, l'autre comme un vecteur.

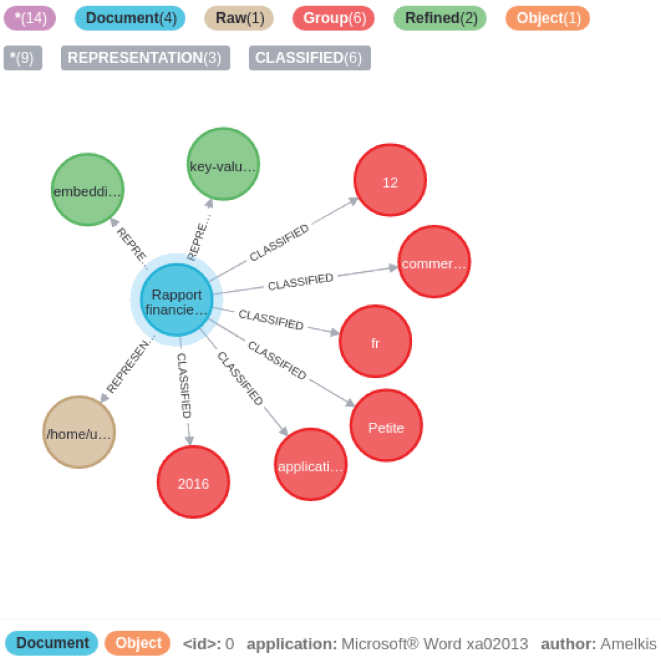

FIGURE 3.4 – Exemple de métadonnées d'un document textuel dans AUDAL

Modèle physique des métadonnées inter Le deuxième niveau du système de métadonnées d'AUDAL gère les relations entre les objets, c'est-à-dire les métadonnées inter. Deux types de relations sont utilisés conformément aux concepts de MEDAL : les regroupements et les liaisons de similarité. Certains des regroupements concernent à la fois les données tabulaires et textuelles, par exemple les regroupements sur le format des données ou encore leur source. À l'inverse, d'autres ne sont pertinents que pour un seul type de données, par exemple le regroupement sur la langue des documents. Les regroupements sont matérialisés dans Neo4j par un ensemble de nœuds. Chaque regroupement est un simple nœud avec lequel tous les objets associés sont liés. Pour illustrer, les six nœuds colorés en rouge dans la figure 3.4 sont des collections de regroupements auxquelles l'objet présenté appartient.

Deux types de liaisons de similarité sont définis en fonction du type de données auquel ils se rapportent. Les *liaisons de similarité entre documents* expriment à quel point deux documents textuels sont similaires. Ces liaisons sont matérialisées par des arêtes non orientées entre des nœuds d'objets dans Neo4j. De même, les liens entre les données tabulaires sont exprimées avec les *liaisons de joignabilité entre tables*. Ces liaisons représentent en fait des dépendances fonctionnelles détectées automatiquement entre les colonnes de différentes tables. Dans Neo4j, les arêtes de joignabilité entre tables sont orientées.

La figure 3.5 illustre un exemple de liaison de joignabilité entre tables. Les deux nœuds bleus représentent deux objets (ici, des tables). Les deux nœuds roses représentent des colonnes contenues dans les tables, et ces nœuds sont ici reliés par des arêtes étiquetées `PK_FK_LINK` pour matérialiser la liaison de joignabilité entre les tables.

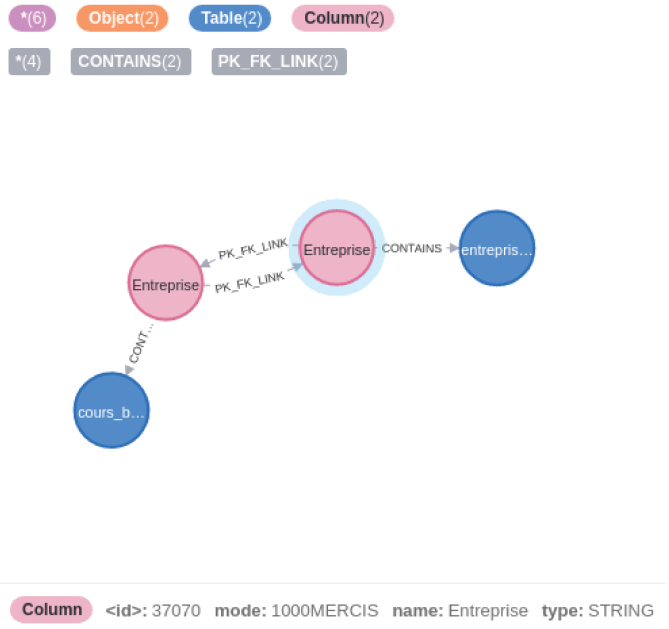

FIGURE 3.5 – Exemple de métadonnées d'une table dans AUDAL

Modèle physique des métadonnées globales Enfin, le troisième niveau du modèle est constitué de métadonnées globales, utilisées pour accélérer ou améliorer les analyses. Il comprend des index qui permettent et accélèrent la recherche par mots-clés sur des documents textuels, ainsi que sur des fichiers de tableur. Ces index sont gérés par ElasticSearch. En outre, le système de métadonnées d'AUDAL comprend également des ressources sémantiques, c'est-à-dire des dictionnaires et des thésaurus. Ces ressources, stockées dans MongoDB, permettent entre autres l'extension automatique des requêtes.

3.4 Évaluation de MEDAL

La section précédente nous a permis d'introduire notre modèle de métadonnées intitulé MEDAL, qui se base sur la typologie de métadonnées présentée dans la section 3.2. Nous avons présenté son modèle conceptuel (représenté graphiquement par un diagramme de classes UML), son modèle logique qui s'appuie la théorie des graphes, et un modèle physique implémenté dans un cas d'usage réel. Nous souhaitons maintenant mettre notre modèle à l'épreuve, pour confirmer son intérêt, mais également percevoir ses faiblesses.

Pour cela, nous commençons par présenter dans la section 3.4.1 comment le modèle MEDAL prend en charge les six fonctionnalités clés qu'un système de métadonnées devrait proposer, fonctionnalités que nous avons identifiées dans le chapitre 2. Nous poursuivons ensuite en comparant les concepts de MEDAL à ceux des modèles de métadonnées les plus récents (section 3.4.2). À la lumière de ces comparaisons, la section 3.4.3 discute des limites de MEDAL, et introduit le besoin de plus de généricité pour un modèle de métadonnées.

3.4.1 Prise en charge des fonctionnalités des systèmes de métadonnées

Grâce à ses concepts, le modèle MEDAL permet de prendre en charge les six fonctionnalités d'un système de métadonnées que nous avons identifiées dans l'état de l'art (chapitre 2). Nous détaillons ci-dessous comment chaque fonctionnalité est prise en charge par les différents concepts de MEDAL.

1. L'**enrichissement sémantique** (ES) fait référence à la capacité d'enrichir les données brutes avec des informations sémantiques. MEDAL permet cela à la fois à travers les attributs assignés aux différents concepts (objet, version, représentation) qui peuvent être par exemple des descriptions, mais aussi grâce aux métadonnées globales, par exemple en utilisant une ressource sémantique (ontologie, thésaurus, ...).
2. L'**indexation des données** (ID) permet de retrouver les données facilement, généralement en s'appuyant sur des recherches par mots-clés. Dans MEDAL, les index et index inversés peuvent être modélisés par des métadonnées globales.
3. Concernant la **génération et conservation de liaisons** (GL), cette fonctionnalité a pour but de lier des données entre elles au sein du système de métadonnées. MEDAL permet plusieurs liens : à l'intérieur d'un objet (mise à jour, transformation) et à l'extérieur (relation de parenté, liaison de similarité, ...). De plus, en fonction des ressources sémantiques présentes dans les métadonnées globales du lac, elles peuvent aussi servir à lier des ensembles de données.
4. Le **polymorphisme des données** (PD) consiste à rassembler les ensembles de données selon leur degré de raffinement (par exemple : données brutes, retravaillées, prêtes à l'analyse). MEDAL prend en charge cette fonctionnalité en proposant de créer de nouvelles représentations des données pour un objet, et une transformation associée. De plus, les objets peuvent aussi être croisés entre eux pour en générer de nouveaux avec la relation de parenté.

5. Le **versionnement des données** (VD) désigne la capacité à suivre l'évolution des données. Cette fonctionnalité est prise en charge dans MEDAL grâce aux différentes versions d'un objet, et les mises à jour associées.
6. Finalement, le **suivi d'utilisation** (SU) trace les interactions entre les utilisateurs et le lac de données. C'est grâce aux métadonnées globales et plus spécifiquement aux *logs* que MEDAL prend en charge cette fonctionnalité. Toute action effectuée par un utilisateur lors de son parcours dans le lac est enregistrée dans les *logs*.

Grâce à ces éléments, MEDAL se trouve donc être le système le plus complet parmi ceux proposés dans la littérature. Le tableau 3.1 reprend la comparaison des 15 systèmes de métadonnées que nous avons effectuée dans l'état de l'art (chapitre 2), mais en y ajoutant MEDAL. Rappelons que ce tableau est remis dans le contexte de l'époque où a été menée cette comparaison, et donc que les propositions les plus récentes n'y figurent pas. En plus de prendre en charge les six fonctionnalités, le second intérêt majeur de MEDAL est qu'il est un modèle de métadonnées, c'est-à-dire réutilisable, et non pas dédié à une seule implémentation.

3.4.2 Comparaison de MEDAL avec les modèles de métadonnées les plus récents

La comparaison précédente a été menée avant la proposition de deux modèles de métadonnées qui sont à la fois les plus récents et les plus complets parmi les modèles de métadonnées. Il s'agit du modèle DAMMS [Ravat and Zhao, 2019b] et du modèle HANDLE [Eichler et al., 2020], tous deux présentés dans le chapitre 2. Pour continuer l'évaluation de MEDAL, nous confrontons ses concepts à ceux de ces deux modèles de métadonnées récents. Pour chaque comparaison, nous utilisons un tableau à deux colonnes. La première colonne liste les concepts de MEDAL et la deuxième colonne reprend les concepts correspondants du modèle comparé.

3.4.2.1 MEDAL vs. DAMMS

MEDAL peut gérer presque tous les concepts de DAMMS [Ravat and Zhao, 2019b] (tableau 3.2). Les versions et représentations sont équivalentes au concept de jeu de données (*Dataset*) et toutes ses sous-classes, comme *Datalake_Datasets*. L'objet et ses regroupements permettent de gérer le concept de mot-clé (*Keyword*), tandis que la liaison de similarité correspond directement à la relation (*Relationship*). Le processus (*Process*) de DAMMS modélise nos concepts de transformation, mise à jour et relation de parenté. Finalement, nous pouvons, à l'aide des métadonnées globales, modéliser les concepts d'utilisateur (*user*) et d'accès (*access*).

MEDAL partage ses avantages et ses inconvénients avec DAMMS, dans le sens où MEDAL introduit également des concepts concrets qui peuvent facilement correspondre aux cas d'usage, mais peut être limité face à des problématiques spécifiques qui nécessitent l'introduction d'autres concepts. Nous notons aussi que MEDAL et DAMMS proposent tous deux la fonctionnalité de polymorphisme des données ou zones multiples, mais d'une

Système \ Fonctionnalités	Type	ES	ID	GL	PD	VD	SU	Total
SPAR [Fauduet and Peyrard, 2010]	♦♯	✓	✓	✓			✓	4/6
[Alrehamy and Walker, 2015]	♦	✓		✓				2/6
[Terrizzano et al., 2015]	♦	✓	✓			✓	✓	4/6
Constance [Hai et al., 2016]	♦	✓	✓					2/6
GEMMS [Quix et al., 2016]	◊	✓						1/6
CLAMS [Farid et al., 2016]	♦	✓						1/6
[Suriarachchi and Plale, 2016]	◊				✓		✓	2/6
[Singh et al., 2016]	♦	✓	✓	✓	✓			4/6
[Farrugia et al., 2016]	♦			✓				1/6
GOODS [Halevy et al., 2016]	♦	✓	✓	✓		✓	✓	5/6
CoreDB [Beheshti et al., 2017]	♦		✓				✓	2/6
Ground [Hellerstein et al., 2017]	◊♯	✓	✓			✓	✓	4/6
KAYAK [Maccioni and Torlone, 2018]	♦	✓	✓	✓				3/6
CoreKG [Beheshti et al., 2018]	♦	✓	✓	✓	✓		✓	5/6
[Diamantini et al., 2018]	◊	✓		✓	✓			3/6
MEDAL	◊	✓	✓	✓	✓	✓	✓	6/6
Total		13/16	10/16	9/16	5/16	4/16	8/16	

♦ : Implémentation d'un lac de données ◊ : Modèle de métadonnées
♯ : Modèle ou implémentation assimilable à un lac de données

TABLE 3.1 – Fonctionnalités proposées par les systèmes de métadonnées de lacs de données en incluant MEDAL

MEDAL	**DAMMS**
Version, Représentation	*Dataset*, sous-classes de *dataset*
Objet, Regroupement	*Keyword*
Liaison de similarité	*Relationship*
Transformation, Mise à jour, Relation de parenté	*Process*
Métadonnées globales	*User*, *Access*

TABLE 3.2 – Concepts de MEDAL et DAMMS

manière différente. Dans MEDAL, les données évoluent au sein d'un objet ou génèrent un nouvel objet, tandis que dans DAMMS, il existe plusieurs zones pour entreposer les données aux degrés de raffinement différents.

Cependant, nous constatons que les concepts du modèle de métadonnées DAMMS sont moins nombreux que ceux de MEDAL. Par exemple, le concept de processus généralise trois concepts de MEDAL (transformation, mise à jour, relation de parenté). Ce constat de MEDAL montre que certains concepts pourraient être généralisés.

3.4.2.2 MEDAL vs. HANDLE

Comparons maintenant les concepts de MEDAL à ceux du modèle de métadonnées HANDLE [Eichler et al., 2020] (tableau 3.3). MEDAL peut gérer les concepts de données (*Data*), métadonnées (*Metadata*) et indicateur de zone (*ZoneIndicator*) grâce aux concepts de version et représentation. En outre, l'objet et ses regroupements correspond à la catégorisation (*Categorization*) de HANDLE. Les liaisons de similarité de MEDAL sont équivalentes aux liens (*Link*) de HANDLE. Toutefois, tous les concepts matérialisant l'évolution des données dans MEDAL (transformation, mise à jour et relation de parenté) n'ont pas d'équivalent dans HANDLE. Mais à l'inverse, l'indicateur de granularité ne trouve pas son pareil parmi les concepts de MEDAL.

MEDAL	**HANDLE**
Version, Représentation	*Data*, *Metadata*, *ZoneIndicator*
Objet, Regroupement	*Categorization*
Liaison de similarité	*Link*
Transformation, Mise à jour, Relation de parenté	—
—	*Granularity Indicator*

Table 3.3 – Concepts de MEDAL et HANDLE

La principale différence entre MEDAL et HANDLE est que ce dernier permet de gérer de multiples niveaux de granularité des données. C'est d'ailleurs la seule caractéristique de modélisation que MEDAL ne prend pas en compte. Toutefois, MEDAL se distingue de HANDLE par sa capacité à versionner les données (à travers les différentes versions et leurs mises à jour), ce que le modèle de Eichler et al. ne permet pas.

3.4.3 Limites de MEDAL

Dans l'état de l'art (chapitre 2), nous avons identifié dans la littérature six fonctionnalités principales que devrait idéalement proposer le système de métadonnées d'un lac de données. Ces travaux portaient sur les systèmes de métadonnées au sens large, et pas spécifiquement sur les modèles de métadonnées, ce qui fait que nous avions étudié

notamment des implémentations de lacs de données qui manquaient parfois de détails sur l'organisation conceptuelle des métadonnées.

Dans des travaux récents, Eichler et al. ont, quant à eux, identifié trois autres caractéristiques [Eichler et al., 2020] pour évaluer des modèles de métadonnées de lacs de données. Cette fois-ci, leur réflexion s'est portée exclusivement sur les modèles de métadonnées, et non les systèmes. Nous listons les trois caractéristiques qu'ils ont identifiées.

Les **propriétés des métadonnées** sont essentielles pour décrire au mieux les données présentes dans le lac. Il est nécessaire que ces propriétés soient capables d'enregistrer des informations variées, car toutes les données ne sont pas exploitées de la même manière par les utilisateurs ou les applications. Les auteurs illustrent ces propriétés des métadonnées en souhaitant enregistrer les interactions d'un utilisateur avec un ensemble de données, et il est alors nécessaire de créer des propriétés adéquates au sein des métadonnées. Bien que cet exemple se rapproche plutôt de la fonctionnalité de suivi d'utilisation que nous avons proposée, cette caractéristique prend tout son sens dans le contexte des métadonnées d'un lac de données.

Les **métadonnées de zone** permettent de connaître le degré de raffinement des données du lac. D'abord données brutes, elles peuvent ensuite être retravaillées pour devenir des données de traitement, puis enfin des données de confiance. Aussi, il est possible d'affecter des métadonnées différentes aux données en exploitant leur degré de raffinement. Par exemple, des données prêtes à l'analyse sont plus susceptibles d'être lues par un plus grand nombre d'utilisateurs que des données brutes. Ces utilisateurs peuvent en outre ne pas avoir les mêmes autorisations en ce qui concerne la lecture de données sensibles. Il est alors envisageable de créer des métadonnées spécifiques pour les données traitées, pour désigner qui a le droit de consulter quoi, par exemple.

Enfin, la prise en charge de **plusieurs niveaux de granularité** vise à pouvoir distinguer la finesse des données présentes dans le lac. Ceci permet alors de savoir si l'ensemble de données considéré est, si nous nous plaçons par exemple dans le cadre des données structurées, un n-uplet, une table ou bien une base de données. En reprenant l'exemple de l'enregistrement des interactions entre un utilisateur et le lac, il est pertinent de savoir si un utilisateur consulte seulement une ligne d'une table ou bien la table entière.

Nous proposons d'ajouter aux six caractéristiques que nous avions initialement identifiées ces trois caractéristiques relevées par [Eichler et al., 2020]. Cependant, au lieu de simplement réunir les caractéristiques, nous fusionnons le polymorphisme des données avec les métadonnées de zone, car ces caractéristiques font toutes deux référence à la même idée. Nous avons également divisé la génération de liens en deux nouvelles fonctionnalités, à savoir les liaisons de similarité et la catégorisation, car certains modèles de métadonnées ne prennent en charge qu'une seule de ces fonctionnalités. Finalement, nous omettons l'indexation des données dans cette comparaison, considérant que l'indexation n'induit pas réellement de problèmes de modélisation des métadonnées. Bien que l'indexation soit certainement pertinente pour évaluer les systèmes de métadonnées, cette fonctionnalité semble moins adaptée aux modèles de métadonnées.

Au total, nous obtenons donc une liste de huit caractéristiques qui peuvent servir à

comparer les modèles de métadonnées des lacs de données et à évaluer leur généricité :

1. Enrichissement sémantique (ES)
2. Polymorphisme des données / zones multiples (PZ)
3. Versionnement des données (VD)
4. Suivi d'utilisation (SU)
5. Catégorisation (CG)
6. Liaisons de similarité (LS)
7. Propriétés de métadonnées (PM)
8. Niveaux de granularité multiples (GM)

Un modèle de métadonnées générique devrait s'adapter à tous les scénarios d'utilisation des lacs de données. Comme chaque cas d'usage nécessite des fonctionnalités spécifiques de gestion des métadonnées, nous considérons que plus les fonctionnalités prises en charge par un modèle de métadonnées sont nombreuses, plus il est générique. Par conséquent, les fonctionnalités sont un moyen approprié de comparer les modèles de métadonnées.

Le tableau 3.4 met en évidence les caractéristiques prises en charge par les cinq modèles examinés dans la section 2.4.2, et y ajoute MEDAL. Cette comparaison montre que les modèles les plus complets sont bien les plus récents, à savoir HANDLE [Eichler et al., 2020], DAMMS [Ravat and Zhao, 2019b], ainsi que notre modèle MEDAL, qui prennent en charge 7 des 8 caractéristiques identifiées. La seule fonctionnalité non prise en charge par HANDLE est le versionnement des données, tandis que DAMMS et MEDAL ne permettent pas d'avoir des niveaux de granularité différents. Après ces deux modèles, c'est Ground [Hellerstein et al., 2017] qui se distingue avec 5 caractéristiques prises en charge. Il est d'ailleurs intéressant de rappeler que Ground n'est pas conçu pour les lacs de données à la base, mais qu'il est tout de même assez pertinent pour être employé dans ce cadre. Rappelons que pour cette comparaison, nous nous contentons de compter le nombre de fonctionnalités prises en charge, sans évoquer l'importance de telle ou telle fonctionnalité.

Sur le plan des caractéristiques, nous constatons que l'enrichissement sémantique (ES) est systématiquement pris en charge par les modèles de métadonnées. Dans une moindre mesure, la catégorisation (CG) et les propriétés de métadonnées (PM) sont gérées par 5 modèles sur 6, le seul ne les proposant pas étant celui de Diamantini et al. [Diamantini et al., 2018]. Il est donc intéressant de noter que ces trois caractéristiques ont été identifiées par les différents auteurs comme primordiales pour un modèle de métadonnées. À l'inverse, les fonctionnalités les moins proposées sont celles du versionnement des données (VD) et des niveaux de granularité multiples (GM), puisque seuls trois modèles les proposent. Toutes les autres fonctionnalités sont proposées par 4 modèles sur 6.

Cette comparaison basée sur les caractéristiques montre qu'aucun des modèles de métadonnées existants n'est suffisamment générique pour répondre à toutes les caractéristiques de modélisation des métadonnées que nous avons identifiées. En particulier,

Modèles \ Caractéristiques	ES	PZ	VD	SU	CG	LS	PM	GM	Total
GEMMS [Quix et al., 2016]	✓				✓		✓	✓	4/8
Ground [Hellerstein et al., 2017]	✓		✓	✓	✓		✓		5/8
[Diamantini et al., 2018]	✓	✓				✓		✓	4/8
DAMMS [Ravat and Zhao, 2019b]	✓	✓	✓	✓	✓	✓	✓		7/8
HANDLE [Eichler et al., 2020]	✓	✓		✓	✓	✓	✓	✓	7/8
MEDAL	✓	✓	✓	✓	✓	✓	✓		7/8
Total	6/6	4/6	3/6	4/6	5/6	4/6	5/6	3/6	

TABLE 3.4 – Caractéristiques prises en charge par les modèles de métadonnées des lacs de données

notre modèle MEDAL ne peut pas prendre en compte de multiples niveaux de granularité. Ce constat nous pousse à vouloir faire évoluer le modèle MEDAL.

Une seconde motivation pour faire évoluer MEDAL provient des travaux menés au laboratoire ERIC sur les différents projets nécessitant la mise en place d'un lac de données. Dans MEDAL, les éléments de données sont considérés soit comme des données brutes, des versions ou des représentations dérivées de données brutes. Les concepts de version et de représentation sont utilisés pour exprimer les données mises à jour et transformées, respectivement. Cependant, en modélisant les métadonnées des trois cas d'usage au sein du laboratoire, nous avons constaté que de nouveaux éléments de données étaient nécessaires, comme par exemple les représentations temporelles. Nous pouvons supposer que de nombreux autres types d'éléments de données sont possibles, d'où un besoin de généralisation.

De même, les mises à jour et les transformations de MEDAL servent à suivre le lignage des représentations et des versions, respectivement ; et les relations de parenté expriment les fusions de données. Tous ces éléments sont liés à des opérations appliquées à des éléments de données. Il peut y avoir d'autres opérations de ce type en fonction des cas d'utilisation particuliers. Finalement, MEDAL présente des liens de similarité, mais d'autres liens sont tout à fait possibles, par exemple des liens hiérarchiques. Là encore, la généralisation est possible.

En prenant du recul, nous pouvons observer que certains concepts de MEDAL sont des concepts concrets, comme par exemple les représentations de données, les transformations ou les liens de similarité. À l'inverse, d'autres concepts, comme les regroupements, sont plus « méta ». À notre avis, MEDAL se situe donc à mi-chemin entre un modèle et un métamodèle. D'ailleurs, DAMMS et HANDLE se situent également, bien que dans une moindre mesure, dans ce niveau d'abstraction de modélisation intermédiaire.

DAMMS présente en effet une hiérarchie de jeux de données dont les niveaux les plus fins sont les jeux de données structurés et semi ou non structurés. Ce choix de modélisation est discutable, d'une part car les données semi-structurées et non structurées sont

rassemblées au sein d'un seul concept, alors que selon nous, les données semi-structurées présentent plus de similitudes avec les données structurées (présence d'attributs et de schéma). D'autre part, la modélisation ne tient pas compte d'autres dimensions possibles des données, telles que les données de flux produites en temps réel, les séries temporelles ou les données spatiales. De même, le modèle de base de HANDLE présente un concept de zone, alors qu'il existe d'autres architectures de lac de données telles que les bassins de données, les architectures fonctionnelles et les architectures basées sur la maturité des données [Sawadogo and Darmont, 2021]. De plus, HANDLE a besoin d'extensions pour préciser ses concepts de granularité, de zone et de catégorisation. Nous imaginons que les concepts de liens et de données pourraient avoir besoin de telles extensions, par exemple pour gérer les liens de similarité et les données de flux, respectivement.

En conclusion, nous faisons face à un compromis où la première option nous permet de définir un modèles de métadonnées comportant des concepts concrets faciles à instancier, mais qui peuvent ne pas être suffisants ou pratiques pour modéliser la plupart des cas d'utilisation. La seconde option consiste à définir un métamodèle de métadonnées, comportant des concepts génériques mais simples, qui conviennent à la plupart des cas d'utilisation. Toutefois, ceux-ci nécessitent un effort de conception plus important pour spécialiser les concepts « méta » en concepts concrets qui peuvent, à leur tour, être instanciés.

Ce deuxième constat nous pousse à emprunter la seconde voie, et donc éviter de rester dans un entre-deux avec un mélange de concepts « méta » et concrets. C'est pourquoi nous avons fait évoluer MEDAL vers plus d'abstraction pour proposer un métamodèle de métadonnées, baptisé goldMEDAL. La présentation de goldMEDAL fait l'objet de la section suivante.

3.5 Présentation du métamodèle goldMEDAL

Un métamodèle de métadonnées peut être exprimé « sous la forme d'un schéma explicite, d'une définition formelle ou d'une description textuelle » [Eichler et al., 2020]. Dans ce chapitre, nous choisissons une approche formelle pour décrire goldMEDAL. Cependant, nous fournissons également un diagramme de classes UML donnant une modélisation semi-formelle, comme pour le modèle MEDAL et les autres modèles de l'état de l'art présentés dans le chapitre 2. De plus, nous utilisons une approche conventionnelle de modélisation des données qui s'appuie sur un métamodèle conceptuel, un métamodèle logique et un modèle physique, afin de démontrer le processus réel de mise en œuvre de notre modèle de métadonnées.

La section 3.5.1 présente les métamodèles conceptuels formels et semi-formels de goldMEDAL. Dans la section 3.5.2, nous détaillons la traduction des concepts de goldMEDAL en un métamodèle logique basé sur la théorie des graphes. Par souci de clarté, les exemples que nous utilisons sont les mêmes exemples dans les deux sections, c'est-à-dire que les exemples au niveau conceptuel sont traduits au niveau logique. Enfin, nous présentons dans la section 3.5.3 deux exemples d'instanciation du métamodèle logique en modèle physique, implémentés avec des technologies différentes (l'un d'eux ayant été developpé

dans le cadre d'un cas d'usage réel).

3.5.1 Métamodèle conceptuel

Le métamodèle de métadonnées goldMEDAL adopte un niveau d'abstraction élevé, pour être à la fois très compréhensible et flexible. Cela contraste avec les modèles de métadonnées précédents qui sont plus prescriptifs [Sawadogo et al., 2019b, Ravat and Zhao, 2019b]. Pour ce faire, goldMEDAL utilise uniquement quatre concepts pour représenter et organiser les données et les métadonnées dans les lacs de données : *entité de données*, *groupement*, *lien* et *processus*. Tous ces concepts sont caractérisés par des attributs ou des propriétés qui constituent leurs propres métadonnées.

3.5.1.1 Entité de données

Les entités de données constituent les unités de base du métamodèle. Elles sont flexibles en termes de granularité des données. Par exemple, une entité de données peut représenter un fichier de tableur, un document textuel ou semi-structuré, une image, une table de base de données, un n-uplet ou une base de données entière. L'introduction de tout nouvel élément dans le lac de données entraîne la création d'une nouvelle entité de données. Il est important d'insister sur le fait que le niveau de granularité de l'entité de données dépend des besoins de l'utilisateur, et donc pourrait être amené à évoluer si nécessaire.

Définition 4 *L'ensemble des entités de données est noté* $\mathcal{E} = \{e_i\}_{i \in \mathbb{N}^*}$.

Notons que dans la présentation de la typologie de métadonnées (section 3.2), nous avions introduit la notion d'objet pour désigner un ensemble homogène de données. Toutefois, un objet peut être composé de plusieurs versions et/ou représentations des données. Pour éviter toute confusion dans la présentation du métamodèle, nous délaissons la notion d'objet pour utiliser le concept d'entité de données.

3.5.1.2 Groupement

Un groupement est une catégorisation d'entités de données. Un groupement est un ensemble de groupes, où un groupe rassemble des entités de données sur la base de propriétés communes. Par exemple, les différentes zones qui sont courantes dans les architectures de lacs de données peuvent être considérées comme les groupes d'un groupement « zone ». Il est possible de considérer que les groupes d'un groupement sont l'équivalent des parties d'un ensemble.

Définition 5 *L'ensemble des groupements est noté* $\mathcal{G} = \{G_j\}_{j \in \mathbb{N}^*}$, *avec* $G_j = \{\Gamma_{jk}\}_{k \in \mathbb{N}^*}$ *et* $\Gamma_{jk} \subseteq \mathcal{E}$ *est un groupe.*

Exemple 1 *Considérons un lac de données architecturé en trois zones : une zone de données brutes, une zone de données retravaillées et une zone de données prêtes à l'analyse. Il*

s'agit d'un groupement que nous désignons par $G_1 = \{\Gamma_{11}, \Gamma_{12}, \Gamma_{13}\}$*, où les groupes* Γ_{11}*,* Γ_{12} *et* Γ_{13} *désignent respectivement les zones de données brutes, retravaillées et prêtes à l'analyse. De plus, supposons que les données peuvent être exprimées en langues française et anglaise. Nous désignons ce second groupement par* $G_2 = \{\Gamma_{21}, \Gamma_{22}\}$*, où les groupes* Γ_{21} *et* Γ_{22} *font référence au français et à l'anglais, respectivement. Enfin, l'ensemble des groupements est noté* $\mathcal{G} = \{G_1, G_2\}$*. Notons que* G_1 *est une partition, ce qui signifie que les entités de données doivent appartenir soit à* Γ_{11}*, soit à* Γ_{12}*, soit à* Γ_{13}*. Inversement, les entités de données de* G_2 *peuvent appartenir à la fois à* Γ_{21} *et à* Γ_{22} *lorsque les deux langues sont utilisées simultanément, ou à aucun des deux si le document ne contient pas de texte.*

Pour illustrer, soit quatre entités de données $\{e_i\}_{i\in[1,4]}$*.* e_1 *et* e_3 *sont des données brutes, tandis que* e_2 *and* e_4 *sont des données retravaillées. Ainsi,* $\Gamma_{11} = \{e_1, e_3\}$ *et* $\Gamma_{12} = \{e_2, e_4\}$*. En outre,* e_1*,* e_2 *et* e_3 *sont des documents en français et* e_4 *est un document en anglais, donc* $\Gamma_{21} = \{e_1, e_2, e_3\}$ *et* $\Gamma_{22} = \{e_4\}$*.*

3.5.1.3 Lien

Les liens sont utilisés pour associer soit des entités de données entre elles, soit des groupes entre eux. Ils peuvent être orientés ou non. Ils servent principalement à exprimer de simples liens de similarité entre entités de données, ou des hiérarchies entre groupes. Par exemple, une hiérarchie temporelle mois $\rightarrow$ trimestre aurait les mois de janvier, février et mars liés au premier trimestre d'une année donnée.

Définition 6 *L'ensemble des liens est noté* $\mathcal{L} = \{l_m\}_{m\in\mathbb{N}^*}$*, avec soit :*

- $l_m : \mathcal{E} \rightarrow \mathcal{E}$*,*
- $l_m : G_j \rightarrow G_{j'}$ *et* $j \neq j'$*.*

Pour illustrer ce concept, nous présentons deux exemples. L'exemple 2 présente un lien entre entités de données, tandis que l'exemple 3 donne une hiérarchie entre groupements.

Exemple 2 *Considérons un lac de données contenant des documents textuels qui doivent être liés pour permettre, par exemple, de trouver des documents similaires ou de regrouper des documents. De telles analyses avancées nécessitent de calculer et de stocker une mesure de similarité, telle que la similarité Cosinus [Allan et al., 2000], pour chaque paire de documents. Pour modéliser la similarité entre deux documents textuels, c'est-à-dire deux entités de données* e_5 *et* e_6*, la mesure de similarité est stockée dans le lien* l_1*, avec* $e_5 \underset{l_1}{\rightarrow} e_6$*. Notons que* l_1 *n'est pas orienté dans ce cas précis, puisque la similarité en cosinus est commutative. Ainsi,* $e_6 \underset{l_1}{\rightarrow} e_5$ *est également valable.*

Une notation fonctionnelle peut également être utilisée : $l_1(e_5) = e_6$ *et* $l_1(e_6) = e_5$*.*

Exemple 3 *Élaborons l'exemple de la hiérarchie entre les mois et les trimestres. Soit* $G_3 = \{\Gamma_{Jan}, \Gamma_{Fev}, ..., \Gamma_{Dec}\}$ *un groupement de mois et* $G_4 = \{\Gamma_{T1}, \Gamma_{T2}, \Gamma_{T3}, \Gamma_{T4}\}$ *un groupement de trimestres. Explicitons maintenant certaines entités de données et leurs*

groupements : $\Gamma_{Jan} = \{e_7, e_8\}$, $\Gamma_{Fev} = \{e_9\}$, $\Gamma_{Mar} = \{e_{10}\}$; $\Gamma_{T1} = \{e_7, e_8, e_9, e_{10}\}$. *Le lien* l_2 *matérialise le lien hiérarchique entre les groupes de* G_3 *et* G_4 *:* $\Gamma_{Jan} \underset{l_2}{\longrightarrow} \Gamma_{T1}, \Gamma_{Fev} \underset{l_2}{\longrightarrow} \Gamma_{T1}, \Gamma_{Mar} \underset{l_2}{\longrightarrow} \Gamma_{T1}$. *Inversement,* $\Gamma_{T1} \underset{l_2^{-1}}{\longrightarrow} \{\Gamma_{Jan}, \Gamma_{Fev}, \Gamma_{Mar}\}$.

Une notation fonctionnelle peut également être utilisée : $l_2(\Gamma_{Jan}) = \Gamma_{T1}$, $l_2(\Gamma_{Fev}) = \Gamma_{T1}$, $l_2(\Gamma_{Mar}) = \Gamma_{T1}, l_2^{-1}(\Gamma_{T1}) = \{\Gamma_{Jan}, \Gamma_{Fev}, \Gamma_{Mar}\}$. *Notons aussi que* $\Gamma_{T1} = \Gamma_{Jan} \cup \Gamma_{Fev} \cup \Gamma_{Mar}$.

3.5.1.4 Processus

Un processus désigne toute transformation appliquée à un ensemble d'entités de données qui produit un nouvel ensemble d'entités de données.

Définition 7 *L'ensemble des processus est noté* $\mathcal{P} = \{P_n\}_{n \in \mathbb{N}^*}$, *avec* $P_n = \{I_n, O_n\}$, $I_n \subseteq \mathcal{E}$ *est l'ensemble des entités de données d'entrée de* P_n *et* O_n *est l'ensemble des entités de données de sortie qui est intégré à* $\mathcal{E}$ *(*$\mathcal{E} \leftarrow \mathcal{E} \cup O_n$*).*

Exemple 4 *Pour illustrer ce concept, soit deux entités de données* e_{11} *et* e_{12} *représentant chacune un fichier HDFS, et* e_{13} *une autre entité de données représentant une table Hive créée à partir de la jointure de* e_{11} *et* e_{12}. *Le processus* P_1 *matérialise la création de* e_{13} *:* $P_1 = \{I_1, O_1\}$, *avec* $I_1 = \{e_{11}, e_{12}\}$ *et* $O_1 = \{e_{13}\}$.

3.5.1.5 Diagramme de classes UML

La figure 3.6 représente un diagramme de classes UML illustrant le métamodèle conceptuel de goldMEDAL. Tous les concepts de goldMEDAL, y compris le groupe, sont modélisés comme des classes (entité de données, groupement, groupe et processus) ou des classes d'association (lien entre entités et lien entre groupes, qui sont respectivement étiquetés Lien-E et Lien-G dans la figure 3.6).

Enfin, bien qu'ils ne soient pas représentés sur la figure 3.6, toutes les classes et classes d'association portent des attributs qui modélisent les propriétés des métadonnées. Ces attributs peuvent être de n'importe quel type, y compris des listes, et varient bien sûr en fonction des cas d'usage.

3.5.2 Métamodèle logique

Tout comme MEDAL, nous avons choisi de représenter le métamodèle logique de goldMEDAL sous la forme d'un graphe, qui est particulièrement bien adapté à la représentation des relations entre les différents concepts.

Ainsi, dans cette section, nous traduisons les concepts définis dans la section 3.5.1 en nœuds, arêtes et hyperarêtes de graphes. De plus, nous illustrons la traduction avec les exemples utilisés au niveau conceptuel. Enfin, nous proposons également une illustration graphique du métamodèle logique de goldMEDAL.

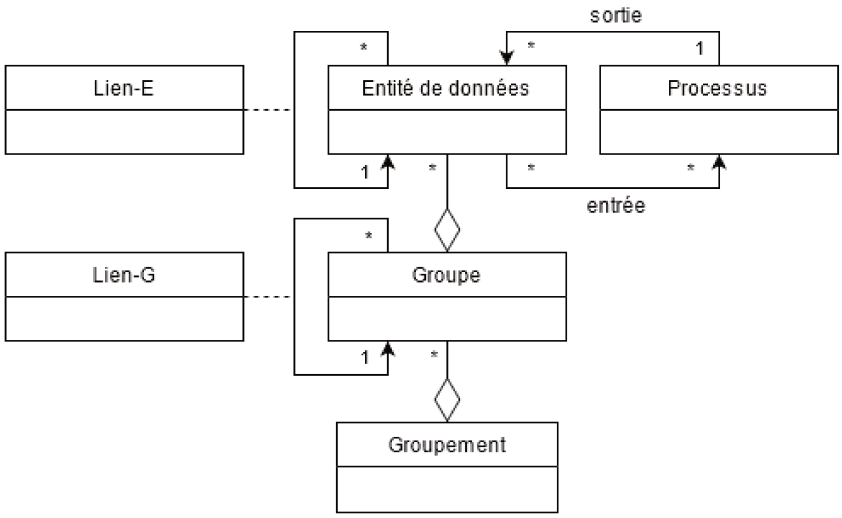

FIGURE 3.6 – Diagramme de classes UML du modèle conceptuel de goldMEDAL

3.5.2.1 Traduction d'entité de données

Les entités de données sont modélisées par des nœuds qui portent des attributs.

Définition 8 *L'ensemble des nœuds est noté* $\mathcal{N} = \{n_i\}_{i \in \mathbb{N}^*}$. *Chaque nœud* $n_i \in \mathcal{N}$ *est porteur d'attributs.*

3.5.2.2 Traduction de groupement

Un groupe est représenté par une hyperarête non orientée, c'est-à-dire une arête qui peut relier plus de deux nœuds. Un groupement est modélisé par un ensemble d'hyperarêtes.

Définition 9 *L'hyperarête (groupe)* k *du groupement* j *est notée* $\theta_{jk} \subseteq \mathcal{N}$, *avec* $j, k \in \mathbb{N}^*$. *Tout* θ_{jk} *est porteur d'attributs.*

Définition 10 *L'ensemble des hyperarêtes du groupement* j *est noté* $H_j = \{\theta_{jk}\}$ *et porte des attributs. L'ensemble des ensembles d'hyperarêtes (l'ensemble des groupements) est noté* $\mathcal{H}$.

Exemple 5 *Traduisons l'exemple 1. Le groupement « zone » est* $H_1 = \{\theta_{11}, \theta_{12}, \theta_{13}\}$, *avec* θ_{11}, θ_{12} *et* θ_{13} *les hyperarêtes représentant les zones (groupes) de données brutes, retravaillées et prêtes à l'analyse, respectivement. Le groupement linguistique est* $H_2 = \{\theta_{21}, \theta_{22}\}$, *avec* θ_{21} *et* θ_{22} *les hyperarêtes représentant les groupes français et anglais, respectivement. Finalement, l'ensemble des groupements, c'est-à-dire l'ensemble des ensembles d'hyperarêtes, est* $\mathcal{H} = \{H_1, H_2\}$.

La figure 3.7 fournit une représentation schématique de l'exemple 5, où les nœuds (traduction des entités de données) sont colorés en orange et les groupes de H_1 et H_2 sont colorés en violet et bleu, respectivement. Nous pouvons voir que n_1 et n_3 appartiennent au groupe de données brutes θ_{11}, tandis que n_2 et n_4 sont dans le groupe de données retravaillées θ_{12}. De plus, n_1, n_2 et n_3 font partie du groupe de langue française θ_{21}, et n_4 du groupe de langue anglaise θ_{22}. Le groupe des données d'accès θ_{13} n'est pas représenté sur cette figure.

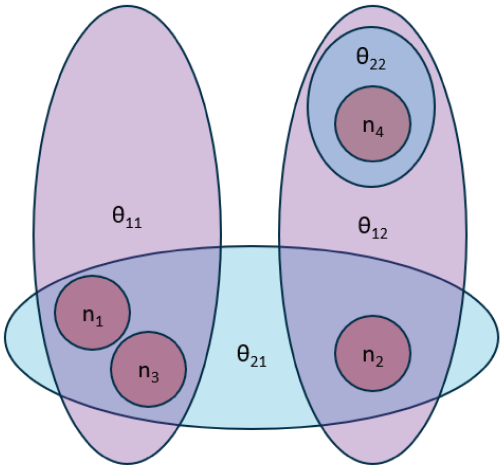

FIGURE 3.7 – Exemple de groupement au niveau logique

3.5.2.3 Traduction de lien

Les liens peuvent modéliser des relations entre des entités de données (nœuds) ou des groupes (hyperarêtes). Ils sont modélisés par des arêtes.

Définition 11 *L'ensemble des arêtes est noté* $\mathcal{A} = \{a_m\}_{m \in \mathbb{N}^*}$*, avec tout* a_m *étant soit :*

- *une arête, orientée ou non, reliant deux nœuds. Alors,* $a_m = (n_i, n_{i'}) \in \mathcal{N}^2$ *;*
- *une arête orientée reliant deux hyperarêtes. Alors,* $a_m = (\theta_{jk}, \theta_{j'k'}) \in H_j \times H_{j'}$*.*

Dans les deux cas, l'arête est porteuse d'attributs.

À nouveau, nous donnons ici deux exemples, pour illustrer à la fois les arêtes entre nœuds (exemple 6) et les arêtes entre hyperarêtes (exemple 7).

Exemple 6 *Continuons maintenant avec la traduction de l'exemple 2 (liaisons de similarité entre documents textuels). Au niveau logique, le lien de similarité entre un couple d'entités de données* n_5 *et* n_6 *est matérialisé par une arête non orientée* $a_1 = (n_5, n_6)$*.*

La figure 3.8 représente graphiquement l'exemple 6.

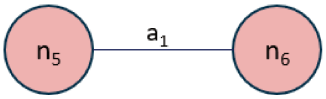

FIGURE 3.8 – Exemple de lien entre entités au niveau logique

Exemple 7 *Pour en revenir à l'exemple de la hiérarchie mois → trimestre de l'exemple 3,* $H_3 = \{\theta_{Jan}, \theta_{Fev}, ..., \theta_{Dec}\}$ *est un ensemble d'hyperarêtes représentant un groupement d'entités de données par mois.* $H_4 = \{\theta_{T1}, \theta_{T2}, \theta_{T3}, \theta_{T4}\}$ *est un ensemble d'hyperarêtes représentant le groupement des trimestres d'une année. Rendons cela explicite avec des instances.* $\theta_{Jan} = \{n_7, n_8\}$, $\theta_{Fev} = \{n_9\}$, $\theta_{Mar} = \{n_{10}\}$; $\theta_{T1} = \{n_7, n_8, n_9, n_{10}\}$. *L'arête* a_2 *matérialise le lien hiérarchique entre* H_3 *et* H_4 *:* $\theta_{Jan} \underset{a_2}{\longrightarrow} \theta_{T1}, \theta_{Fev} \underset{a_2}{\longrightarrow} \theta_{T1}, \theta_{Mar} \underset{a_2}{\longrightarrow} \theta_{T1}$. *Inversement,* $\theta_{T1} \underset{a_2^{-1}}{\longrightarrow} \{\theta_{Jan}, \theta_{Fev}, \theta_{Mar}\}$.

L'exemple 7 est décrit visuellement par la figure 3.9. Les groupes de H_3 (θ_{Jan}, θ_{Fev}, ...) sont représentés en vert, et les groupes de H_4 sont en gris. Le lien a_2 connecte les groupes de H_3 à H_4. Notons que les entités de données présentes dans les différents groupes ne sont pas visibles sur la figure.

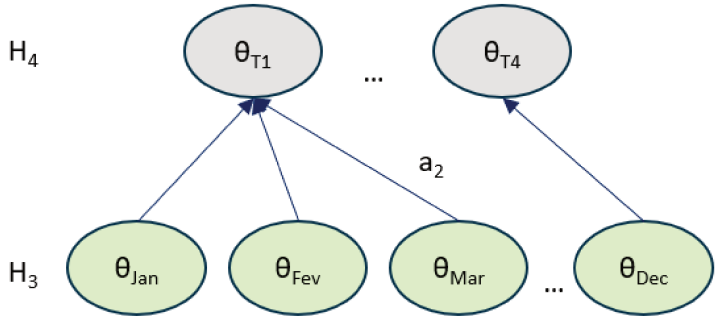

FIGURE 3.9 – Exemple de lien entre groupements au niveau logique

3.5.2.4 Traduction de processus

Un processus est modélisé par une hyperarête orientée.

Définition 12 *L'ensemble des hyperarêtes orientées est noté* $\mathcal{Q} = \{\Pi_n\}_{n \in \mathbb{N}^*}$, *avec* $\Pi_n = \{\Upsilon_n, \Omega_n\}$, $\Upsilon_n \subseteq \mathcal{N}$ *étant l'ensemble des nœuds d'entrée de* Π_n *et* Ω_n *l'ensemble des nœuds de sortie intégrés à* $\mathcal{N}$ *(*$\mathcal{N} \leftarrow \mathcal{N} \cup \Omega_n$*). Tout* Π_n *est porteur d'attributs.*

Exemple 8 *Enfin, traduisons l'exemple 4. L'hyperarête orientée représentant le processus de création d'une table Hive (représentée par le nœud* n_{13}*) à partir de la jointure de deux fichiers HDFS (représentés par les nœuds* n_{11} *et* n_{12}*) est* $\Pi_1 = \{\Upsilon_1, \Omega_1\}$, *avec* $\Upsilon_1 = \{n_{11}, n_{12}\}$ *et* $\Omega_1 = \{n_{13}\}$.

La figure 3.10 donne une représentation graphique de l'exemple 8. L'entrée et la sortie de Π_1, respectivement Υ_1 et Ω_1, sont colorées en jaune. Notons qu'une flèche en pointillés relie Υ_1 à Ω_1, pour indiquer que l'hyperarête Π_1 est orientée.

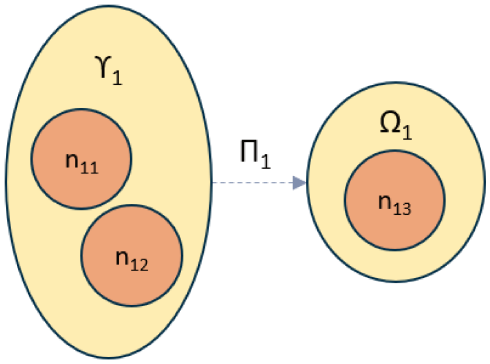

FIGURE 3.10 – Exemple de processus au niveau logique

3.5.3 Modèles physiques

Pour compléter la présentation de goldMEDAL, nous montrons dans cette section comment le métamodèle logique peut être instancié et traduit en un modèle physique. Pour ce faire, nous commençons par montrer une implémentation de goldMEDAL avec le système de gestion de base de données (SGBD) graphe Neo4j [1] (section 3.5.3.1), en reprenant les exemples des sections précédentes. Nous enchaînons par un deuxième exemple qui se base sur un cas d'usage réel, et dont l'implémentation utilise le système de gouvernance des données et de métadonnées Apache Atlas [2] (section 3.5.3.2).

3.5.3.1 Modèle physique avec Neo4j

Ce premier modèle physique de goldMEDAL utilise le SGBD orienté graphe Neo4j, qui est adapté pour représenter les relations. La principale différence entre le métamodèle logique de goldMEDAL et le modèle physique de Neo4j réside dans le fait que Neo4j ne supporte pas les hyperarêtes. Pour pallier ce problème, une solution simple consiste à créer un nœud supplémentaire qui matérialise l'hyperarête, et le connecter à tous les nœuds concernés.

Ainsi, les groupes (et groupements) ainsi que les processus seront matérialisés dans Neo4j en créant des nœuds. En ce qui concerne les liens, puisque ce sont des arêtes au niveau logique, il n'est pas nécessaire de créer un nœud, il est possible de connecter directement les nœuds concernés (entités de données ou groupes), mais cette option reste possible.

1. `https://neo4j.com`
2. `https://atlas.apache.org`

La figure 3.11 illustre une implémentation de l'exemple 5 dans Neo4j. Les nœuds représentant les entités de données sont colorés en orange, et les nœuds représentant les groupements (dans ce cas, Zone et Langue) sont colorés en vert. Les entités de données sont connectées aux nœuds violets (resp. bleus) qui représentent les groupes du groupement Zone (reps. groupement Langue). Les nœuds de groupe sont connectés au nœud de groupement auquel ils appartiennent.

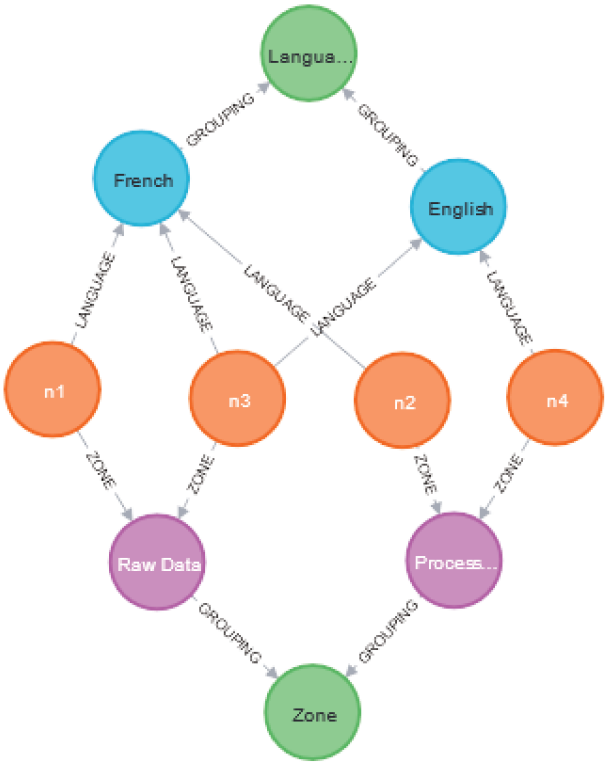

FIGURE 3.11 – Exemple de groupement au niveau physique avec Neo4j

La figure 3.12 traduit deux manières d'implémenter l'exemple 6 dans un modèle physique basé sur Neo4j. À gauche, nous voyons deux nœuds oranges représentant deux entités de données, connectés par une arête pouvant contenir des valeurs en tant qu'attributs (par exemple une mesure de similarité). Précisons que Neo4j ne permet pas de créer des arêtes non orientées, c'est pourquoi nous avons défini une direction arbitraire pour les arêtes de similarité. Cependant, cela n'affecte pas le fait que de telles arêtes peuvent être interrogées comme non orientées. Sur la partie droite de la figure, nous voyons à nouveau les deux nœuds oranges (entités de données), reliés cette fois-ci à un autre nœud rose, qui matérialise le lien. Comme avant, le nœud peut contenir diverses informations, comme la valeur de la mesure de similarité (valant ici 0.67). Remplacer l'arête par un nœud est ainsi une autre approche possible dans Neo4j.

Enfin, la figure 3.13 montre comment l'exemple 8 peut être géré dans Neo4j. Le processus, nommé ici « script1 » est matérialisé par un nœud (coloré en jaune). Les

FIGURE 3.12 – Exemple de lien au niveau physique avec Neo4j

nœuds qui sont l'entrée du processus sont connectés avec une arête allant des entités de données au processus. Inversement, les nœuds de sortie sont connectés avec une arête allant du processus aux entités de données.

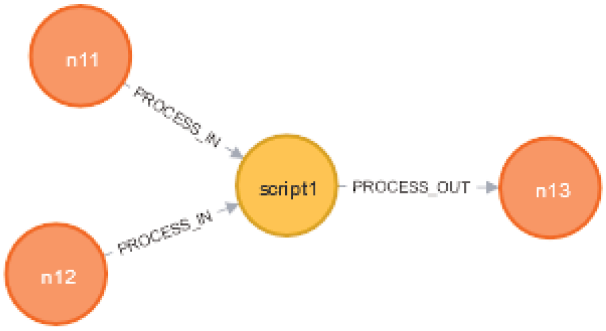

FIGURE 3.13 – Exemple de processus au niveau physique avec Neo4j

En plus de ce modèle physique, nous présentons en détail dans le chapitre 4 un second modèle physique basé sur Neo4j, dans le cadre de notre cas d'usage sur l'habitat social.

3.5.3.2 Modèle physique avec Apache Atlas

Le deuxième exemple s'appuie sur le cas d'usage tiré du projet « Hyper thésaurus et lacs de données : fouiller la ville et ses archives archéologiques » (HyperThésau) [Liu et al., 2021] où un lac, nommé ArchaeoDAL, a été conçu par des chercheurs du laboratoire ERIC avec qui nous avons collaboré pour proposer le métamodèle goldMEDAL. Ce lac de données est dédié aux données archéologiques, qui peuvent être de différents types, par exemple des documents textuels (rapports de fouilles), des images (photographies, dessins, plans...), des données de capteurs, des résultats d'analyses chimiques, etc. En outre, les données structurées sont souvent produites par divers dispositifs qui ne sont pas toujours compatibles entre eux. La description d'un objet archéologique diffère également en fonction des utilisateurs, des usages et du temps. Ainsi, les archéologues utilisent des ressources sémantiques telles que des thésaurus pour faire interagir des données d'origines diverses. Là encore, un lac de données peut aider à stocker ces données, à intégrer des ressources sémantiques et à connecter des données hétérogènes.

Modèle physique des entités de données L'implémentation d'ArchaeoDAL s'appuie fortement sur l'écosystème Apache. En particulier, son système de métadonnées repose sur Apache Atlas. Les objets d'Atlas correspondent aux entités de données de goldMEDAL, et outre les propriétés des métadonnées (sous la forme de paires clé-valeur), ils peuvent également se rapporter à des termes de thésaurus et à des classifications, c'est-à-dire à des groupements goldMEDAL (figure 3.14).

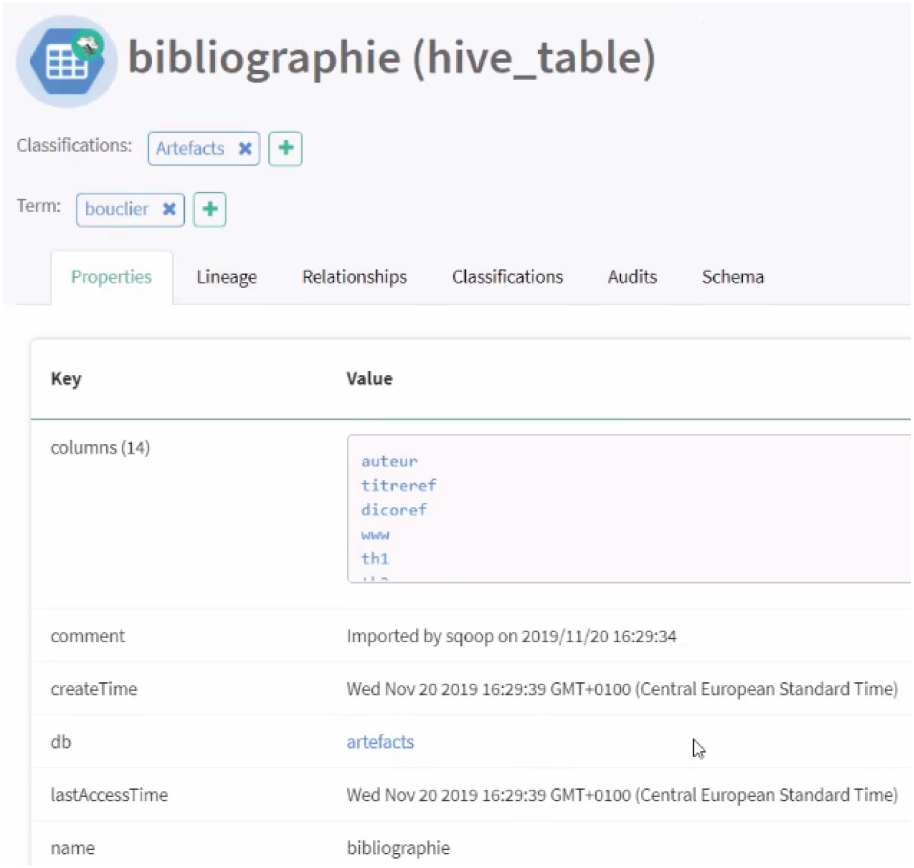

FIGURE 3.14 – Exemple d'objet dans Atlas

En outre, les types d'objets d'Atlas sont exploités pour répondre aux exigences métier spécifiques concernant les propriétés des métadonnées. Par exemple, dans le projet HyperThésau, les utilisateurs ont besoin non seulement de métadonnées sémantiques pour comprendre le contenu des données, mais aussi de métadonnées géographiques pour savoir où les objets archéologiques ont été découverts. Les avantages d'un système de types d'objets sont les suivants :

- consistance : une définition universelle des métadonnées permet d'éviter les variations terminologiques susceptibles de causer des problèmes d'extraction de données ;
- flexibilité : un système de types propres au domaine permet de définir des métadonnées spécifiques pour les exigences de chaque cas d'usage ;

- efficacité : avec un système de types de métadonnées donné, il est facile d'écrire et de mettre en œuvre des requêtes de recherche. Comme les noms et les types de toutes les propriétés des métadonnées sont connus à l'avance, nous pouvons filtrer les données avec des prédicats de métadonnées tels que *upload_date* > '10/02/2016'.

Modèle physique des processus Atlas comprend également une fonctionnalité de lignage intéressante qui permet de visualiser les chaînes de processus. Par exemple, la figure 3.15 représente un simple processus d'ingestion de données brutes stockées dans HDFS dans une table Hive, où les objets sont symbolisés par des hexagones bleus et le processus par un hexagone vert.

FIGURE 3.15 – Exemple de lignage des données dans Atlas

Thésaurus et liens Le projet HyperThésau s'appuie fortement sur les thésaurus pour interroger les données. Dans Atlas, un thésaurus consiste en un ensemble de catégories et de termes qui permettent de regrouper les données. Dans le glossaire d'Atlas, une catégorie ne peut avoir qu'un seul parent ; une catégorie sans parent est appelée catégorie racine. À l'inverse, une catégorie peut avoir plusieurs sous-catégories ou termes. Un terme doit avoir une catégorie parente mais pas de sous-catégorie. Un terme peut avoir des relations (i.e. des liens goldMEDAL) avec d'autres termes, par exemple des mots apparentés, des synonymes, des antonymes, etc. Ainsi, avec le glossaire d'Atlas et donc les catégories et les termes, il serait facile de représenter des ontologies ou des taxonomies.

Finalement, des liens spécifiques sont ajoutés entre les nœuds de données associés aux nœuds de termes du thésaurus. La partie gauche de la figure 3.16 affiche un extrait du thésaurus. La figure 3.16 montre également comment un terme (arme défensive) pointe

vers les métadonnées correspondantes (descriptions courtes et longues) et les termes connexes.

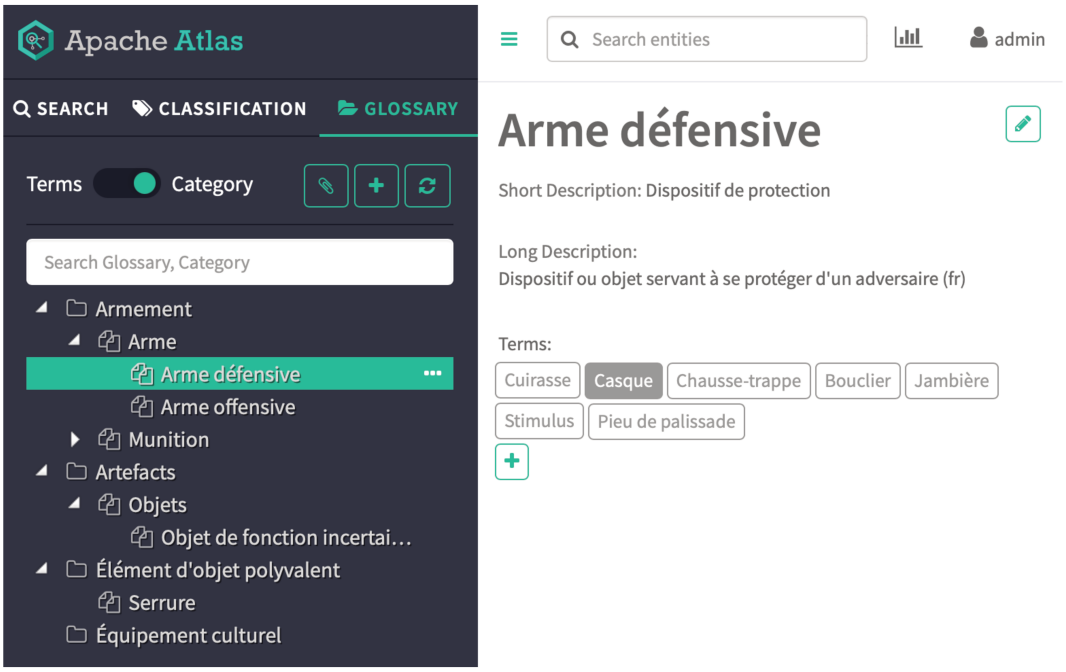

FIGURE 3.16 – Exemple de thésaurus dans Atlas

3.6 Evaluation de goldMEDAL

Après avoir présenté en détail le métamodèle de métadonnées goldMEDAL, nous démontrons dans cette section l'efficacité de notre proposition. À cette fin, pour valider goldMEDAL, nous commençons par étudier dans la section 3.6.1 comment il prend en charge les caractéristiques de modélisation des métadonnées (présentées dans la section 3.4.3). Ensuite, dans la section 3.6.2 nous comparons les concepts de goldMEDAL à ceux des modèles de métadonnées les plus récents, MEDAL y compris.

3.6.1 Prise en charge des caractéristiques de modélisation des métadonnées

Pour évaluer goldMEDAL, nous n'allons pas utiliser les six premières fonctionnalités identifiées pour comparer les systèmes de métadonnées. En effet, goldMEDAL est un métamodèle de métadonnées qui se veut générique ; le comparer à des implémentations sur des fonctionnalités assez techniques (par exemple, l'indexation des données) ne permet pas de faire ressortir sa généricité. C'est pourquoi nous allons le confronter aux huit caractéristiques de modélisation, identifiées dans la section 3.4.3, qui sont dédiées exclu-

sivement aux modèles de métadonnées, et permettent de donner une meilleure évaluation de la généricité d'un modèle.

Les quatre concepts définis dans goldMEDAL permettent de couvrir toutes les caractéristiques de modélisation des métadonnées. Dans ce qui suit, nous détaillons comment chaque fonctionnalité est prise en charge.

1. L'**enrichissement sémantique** (ES) fait référence à la capacité d'enrichir les données brutes avec des informations sémantiques. Ceci est fait dans goldMEDAL par le biais des attributs qui sont assignés aux entités de données. Par exemple, une description ou même une référence à un concept ontologique peut être attribuée aux entités de données, puis aux nœuds dans le métamodèle logique et le modèle physique.
2. Le **polymorphisme des données (ou zones multiples)** (PZ) consiste à organiser les entités de données en fonction de leur niveau de raffinement. Ceci est réalisé dans goldMEDAL en utilisant des groupements. Chaque niveau de raffinement (données brutes, données traitées, etc.) est matérialisé par un groupe dans le groupement « zone ».
3. Le **versionnement des données** (VD) désigne la capacité à suivre l'évolution des données. Cette fonctionnalité est prise en charge par les processus de goldMEDAL, qui servent à suivre les opérations, y compris les modifications appliquées aux données.
4. Le **suivi d'utilisation** (SU) trace les interactions entre les utilisateurs du lac et les données. C'est précisément l'objectif des processus goldMEDAL. Le concept de processus sert en effet à tracer tout le contexte des opérations appliquées aux données, notamment l'utilisateur, l'action effectuée, les données impactées, etc.
5. La **catégorisation** (CG) consiste à organiser les entités de données en collections sur la base d'une caractéristique partagée. Cette fonctionnalité correspond parfaitement au concept de groupement de goldMEDAL.
6. Les **liaisons de similarité** (LS) servent à suivre la similitude entre les entités de données. goldMEDAL prend en charge cette fonctionnalité par le biais du concept de lien, car il sert, entre autres, à connecter les entités de données.
7. Les **propriétés des métadonnées** (PM) sont des informations atomiques associées aux entités de données. Elles sont supportées par goldMEDAL, de la même manière que l'enrichissement sémantique, c'est-à-dire à l'intérieur des entités de données.
8. Les **niveaux de granularité multiples** (GM) permettent de gérer simultanément des données ayant différents niveaux de granularité dans un lac de données. Cette fonctionnalité est également prise en charge dans goldMEDAL par le biais de groupements, c'est-à-dire que chaque niveau de granularité peut être matérialisé par un groupe, qui rassemble toutes les entités de données associées. HANDLE [Eichler et al., 2020] utilise une approche similaire, avec un indicateur de granularité, c'est-à-dire une balise qui exprime les niveaux de granularité des données.

Le tableau 3.5 reprend la comparaison des modèles de métadonnées faite dans l'état de l'art (chapitre 2), mais en y ajoutant notre premier modèle MEDAL et notre méta-modèle goldMEDAL. Nous constatons que le susnommé propose les huit caractéristiques identifiées pour évaluer la généricité des modèles, ce qui en fait donc le métamodèle le plus générique, à notre connaissance.

Modèles \ Caractéristiques	**ES**	**PZ**	**VD**	**SU**	**CG**	**LS**	**PM**	**GM**	**Total**
GEMMS [Quix et al., 2016]	✓				✓		✓	✓	4/8
Ground [Hellerstein et al., 2017]	✓		✓	✓	✓		✓		5/8
[Diamantini et al., 2018]	✓	✓				✓		✓	4/8
[Ravat and Zhao, 2019b]	✓	✓	✓	✓	✓	✓	✓		7/8
HANDLE [Eichler et al., 2020]	✓	✓		✓	✓	✓	✓	✓	7/8
MEDAL	✓	✓	✓	✓	✓	✓	✓		7/8
goldMEDAL	✓	✓	✓	✓	✓	✓	✓	✓	8/8
Total	7/7	5/7	4/7	5/7	6/7	5/7	6/7	4/7	

TABLE 3.5 – Caractéristiques prises en charge par goldMEDAL par rapport aux autres modèles de métadonnées

3.6.2 Comparaison de goldMEDAL avec les modèles de métadonnées les plus récents

Comme pour MEDAL, nous évaluons la complétude des concepts de goldMEDAL en le comparant aux trois modèles de métadonnées qui sont à la fois les plus récents et les plus complets parmi les modèles de métadonnées, c'est-à-dire à nouveau DAMMS et HANDLE (tous deux présentés dans le chapitre 2). Cette fois-ci, nous ajoutons aussi une comparaison à notre premier modèle, MEDAL. À nouveau, pour chaque comparaison, nous utilisons un tableau à deux colonnes. La première colonne liste les concepts de goldMEDAL et la deuxième colonne les concepts correspondants du modèle comparé. Lorsqu'un concept n'a pas d'équivalent, il est marqué par le symbole « — ».

3.6.2.1 goldMEDAL vs. DAMMS

goldMEDAL peut gérer presque tous les concepts du modèle de métadonnées DAMMS [Ravat and Zhao, 2019b] (tableau 3.6). L'entité de données généralise le concept de jeu de données (*Dataset*) et toutes ses sous-classes, telles que *Datalake_Datasets* et *Source_Datasets*. Le groupement généralise le concept de mot-clé (*Keyword*). Enfin, lien et processus correspondent directement à relation (*Relationship*) et processus (*Process*), respectivement.

Cependant, deux concepts de DAMMS, à savoir utilisateur (*user*) et accès (*access*), n'ont pas d'équivalent explicite dans goldMEDAL. Néanmoins, les fonctionnalités sup-

goldMEDAL	**DAMMS**
Entité de données	*Dataset*, sous-classes de *dataset*
Groupement	*Keyword*
Lien	*Relationship*
Processus	*Process*
—	*User*, *Access*

TABLE 3.6 – Concepts de goldMEDAL et DAMMS

portées par ces concepts existent dans goldMEDAL. Ils font en effet référence au suivi de l'utilisation des données, qui est couvert dans goldMEDAL par des processus.

3.6.2.2 goldMEDAL vs. MEDAL

Les quatre concepts principaux de goldMEDAL permettent de généraliser tous les concepts de MEDAL (tableau 3.7). Dans goldMEDAL, le concept d'entité de données couvre les trois concepts spécifiques à MEDAL : données brutes, représentation et version qui représentent les éléments de données. Les concepts goldMEDAL de groupement et de groupe généralisent les concepts MEDAL de regroupement et d'objet. Le lien généralise le concept de liaison de similarité et, enfin, le processus généralise les concepts de transformation, mise à jour et relation de parenté.

goldMEDAL	**MEDAL**
Entité de données	Données brutes, Représentation, Version
Groupement	Regroupement
Groupe	Objet
Lien	Liaison de similarité
Processus	Transformation, Mise à jour, Relation de parenté

TABLE 3.7 – Concepts de goldMEDAL et MEDAL

Notons que nous ne mentionnons pas dans cette comparaison les métadonnées globales qui existent dans MEDAL. En effet, nous pouvons considérer que des éléments tels que les *logs* ou les index induisent principalement des problèmes d'implémentation, plutôt que de modélisation des métadonnées. Cependant, d'autres formes de métadonnées globales, à savoir les ressources sémantiques telles que les thésaurus et les ontologies, peuvent tout à fait être modélisées avec goldMEDAL en utilisant les concepts d'entité de données, de groupement et de lien.

3.6.2.3 goldMEDAL vs. HANDLE

goldMEDAL peut également généraliser les concepts de HANDLE (tableau 3.8). L'entité de données généralise à la fois les données (*Data*) et les métadonnées (*Metadata*), puisqu'une entité de données est une représentation de données qui contient également des propriétés de métadonnées. Le groupement généralise trois concepts : catégorisation (*Categorization*), indicateur de zone (*ZoneIndicator*) et indicateur de granularité (*GranularityIndicator*). Enfin, le processus n'a pas de correspondance directe dans HANDLE, bien que les auteurs montrent que les processus peuvent être modélisés par des instances de métadonnées d'action *Action metadata* provenant de l'extension de catégorisation de HANDLE [Eichler et al., 2020].

goldMEDAL	**HANDLE**
Entité de données	*Data*, *Metadata*
Groupement	*Categorization*, *ZoneIndicator* *GranularityIndicator*
Lien	*Link*
Processus	—

Table 3.8 – Concepts de goldMEDAL et HANDLE

La gestion de multiples niveaux de granularité comme dans HANDLE n'était pas supportée par MEDAL, c'était donc un objectif de conception pour goldMEDAL. Bien qu'il n'y ait pas d'indicateur de granularité explicite dans goldMEDAL, toute entité de données peut avoir une propriété de granularité. Cependant, une manière plus efficace est de définir les entités de données sur le niveau de granularité le plus fin possible. Ensuite, des niveaux de granularité plus grossiers sont obtenus avec des groupements. Par exemple, si chaque entité de données correspond à un n-uplet dans une base de données relationnelle, alors un groupement représente un ensemble de tables – peut-être une base de données entière ; les tables de l'ensemble étant modélisées par des groupes.

3.7 Conclusion

Dans ce chapitre, nous avons présenté une typologie de métadonnées, qui se base sur la notion d'objet pour désigner un ensemble de données, et les métadonnées sont organisées en trois grandes catégories : les métadonnées intra-objet, inter-objets et globales. Les métadonnées intra-objet sont associées à chaque ensemble de données homogènes sous la forme de prévisualisations, de versions, de représentations, de métadonnées sémantiques ou de propriétés. Les métadonnées inter-objets permettent elles de lier les objets entre eux, notamment au sein de collections ou à travers des liaisons de similarité ou de parenté. Enfin, les métadonnées globales servent à faciliter et améliorer les analyses des données ainsi que l'utilisation du lac de données de manière générale.

Au niveau logique, MEDAL organise les métadonnées sous forme de graphes. Un objet est représenté par un hypernœud contenant des nœuds qui correspondent aux versions et représentations de l'objet. Les opérations de transformation et de mise à jour sont modélisées par des arcs orientés reliant les nœuds. Les hypernœuds peuvent être liés de plusieurs manières : des arcs pour modéliser les liaisons de similarité et des hyperarcs pour traduire les groupements et liaisons de parenté. Enfin, les ressources globales sont aussi présentes, sous la forme de bases de connaissances, d'index ou encore de journaux d'événements ; nous nous intéressons peu à leur représentation dans notre modèle de métadonnées car elles se situent à un niveau plus opérationnel et dépendent fortement du support technologique utilisé.

En reprenant les six fonctionnalités d'un système de métadonnées identifiées dans le chapitre 2, nous constatons que MEDAL est le seul système de métadonnées qui propose l'intégralité de ces fonctionnalités. Toutefois, des comparaisons avec des modèles de métadonnées plus récents, notamment HANDLE [Eichler et al., 2020], nous poussent à modifier nos critères d'évaluation. En considérant cette fois huit caractéristiques de généricité, nous faisons le constat que MEDAL, au même titre que les autres modèles, n'est pas suffisamment générique.

C'est pourquoi nous avons ensuite présenté notre métamodèle de métadonnées goldMEDAL, une évolution de MEDAL, qui se veut être plus générique. goldMEDAL est basé sur quatre concepts principaux : entité de données, groupement, lien et processus, qui sont définis aux niveaux conceptuel, logique et physique. Ces concepts interagissent ensemble pour répondre aux exigences de gestion des métadonnées des lacs de données et généralisent tous les concepts précédemment proposés dans les plus récents modèles de métadonnées. Le concept de groupement prend notamment en charge l'organisation des lacs de données en zones [Ravat and Zhao, 2019b]. En combinant les concepts d'entité de données, de groupement et de lien, il est aussi possible de gérer plusieurs niveaux de granularité des données comme dans HANDLE [Eichler et al., 2020].

En outre, goldMEDAL prend en charge les huit caractéristiques identifiées pour comparer les modèles de métadonnées des lacs de données, ce qui en fait le métamodèle de métadonnées le plus générique à notre connaissance. Une autre particularité de goldMEDAL est la possibilité explicite de tracer le lignage des données avec le concept de processus. goldMEDAL gère ainsi la dynamique des données, alors que le modèle de métadonnées le plus récent de la littérature, HANDLE [Eichler et al., 2020], ne le supporte pas nativement.

Le métamodèle goldMEDAL est utilisé pour répondre à un problème métier rencontré au sein de l'entreprise BIAL-X qui nécessite la mise en œuvre concrète d'un lac de données et de son système de métadonnées. Ces travaux seront présentés en détail dans le chapitre 4.

Chapitre 4

HOUDAL, un lac de données dédié à l'habitat social

Sommaire

"Have you ever had a dream Neo, that you were so sure was real ? What if you were unable to wake from that dream ? How would you know the difference between the dream world, and the real world ?"

Morpheus, *The Matrix (1999)*

Résumé

Comme tous les secteurs de l'activité économique, l'habitat social est impacté par l'essor des mégadonnées. Si les analyses d'informatique décisionnelle et de science des données sont plus ou moins maîtrisées par les bailleurs sociaux, l'entreposage et l'exploitation de mégadonnées dépassent parfois les capacités des outils traditionnels, ce qui soulève une importante problématique. Pour faire face à ces problèmes, nous proposons d'utiliser un lac de données, un système dans lequel des données de tous types peuvent être stockées et à partir duquel diverses analyses peuvent être effectuées. La définition de l'architecture fonctionnelle d'un lac de données est une problématique de recherche abordée dans la littérature scientifique. Bien que plusieurs propositions aient été formulées, les architectures ont toutes tendance à compartimenter les données au sein au lac, ce qui est contraire à notre approche.

Dans ce chapitre, nous présentons un cas d'usage réel sur l'habitat social qui a motivé notre volonté d'utiliser un lac de données. Nous proposons également une architecture de lac de données basée sur quatre composants et organisée de manière cyclique, dont l'objectif est de ne pas compartimenter les données du lac. Nous montrons aussi comment nous pouvons instancier puis traduire notre métamodèle de métadonnées goldMEDAL en un modèle physique. Enfin, nous présentons HOUDAL, une implémentation d'un lac de données basé sur notre architecture. Il repose sur un assemblage ad hoc de technologies, en particulier le système de gestion de base de données graphe Neo4j.

4.1 Introduction

Dans le domaine de l'habitat social, l'utilisation des données permet aux bailleurs sociaux d'améliorer la gestion des logements, et celle des locataires. Grâce à l'informatique décisionnelle, ou *Business Intelligence* (BI), les données sont organisées au sein d'un entrepôt de données, dans l'optique de fournir aux décideurs (directeurs d'agence, contrôleurs de gestion, etc.) des tableaux de bord leur permettant de piloter leur business. À l'aide de ces rapports, les décideurs peuvent faire de la gestion locative, financière ou encore surveiller les impayés. Les analyses BI consistent à « regarder en arrière » pour gérer son activité [Watson and Wixom, 2007]. En parallèle, les méthodes de science des données (*Data Science*) appliquées au logement social peuvent être utilisées pour des analyses descriptives, comme l'identification de groupes de locataires ou de logements, mais aussi pour des analyses prédictives, par exemple pour prévoir les impayés ou la vacance d'un logement. Ces analyses permettent de mieux comprendre son activité en « regardant vers l'avant » [Mortenson et al., 2015].

Cependant, au cours des dix dernières années, la révolution des mégadonnées a bouleversé la manière dont les secteurs économiques extraient la connaissance des données [Gandomi and Haider, 2015, Assunção et al., 2015], et le logement public ne fait pas exception. De nouvelles données, souvent non structurées, apparaissent et peuvent être exploitées. C'est notamment le cas des appels téléphoniques (réclamations des locataires), ou des réseaux sociaux. Également, de plus en plus de données ouvertes deviennent disponibles, notamment les données géographiques, à partir desquelles il est possible d'obtenir des informations sur les écoles, les services, ou le taux d'emploi par secteur géographique, par exemple [Gandomi and Haider, 2015, Chen et al., 2014]. Tout cela pose de nouveaux problèmes à résoudre avant d'arriver à l'objectif de pouvoir mener des analyses, simples ou avancées, sur tous types de données, massives ou non.

C'est dans cette optique que le concept de lac de données a fait son apparition [Dixon, 2010, Miloslavskaya and Tolstoy, 2016, Sawadogo et al., 2019b]. Rappelons que le concept est encore relativement récent, mais de nombreux travaux sont actuellement menés par les chercheurs dans la littérature. Parmi ces problèmes de recherche, nous trouvons celui de la définition de l'architecture d'un lac de données, et plusieurs propositions ont été formulées [Sawadogo and Darmont, 2021]. Nous pouvons notamment citer les architectures basées sur la maturité des données mettant en place des zones [Ravat and Zhao, 2019b, LaPlante and Sharma, 2016], ou encore les architectures privilégiant le format des données en proposant des bassins [John and Misra, 2017, Inmon, 2016]. Toutefois, nous constatons que ces architectures, certes très pertinentes, sont souvent adaptées pour un besoin spécifique et influencent fortement l'utilisation du lac de données.

Dans ce chapitre, nous nous appuyons sur un cas d'usage réel, à savoir un projet ayant été mené par des collaborateurs de BIAL-X, l'entreprise dans laquelle se déroule la thèse CIFRE. Il met en évidence les problèmes inhérents au fait de stocker des données aux formats variés et de les analyser par des méthodes diverses, et ce avec des outils traditionnels. Il devient alors très compliqué de suivre le lignage des données et, de manière plus générale, d'avoir une vraie gouvernance des données pour pouvoir décrire

les tenants et aboutissants du projet de manière claire et compréhensible. Ce projet, qui n'a pas été mené au sein d'un lac de données, nous permet de motiver la nécessité d'un lac pour résoudre de manière efficace les problèmes rencontrés.

Nous proposons dans ce chapitre une architecture de lac de données qui vise à ne pas compartimenter les données du lac. Basée sur quatre composants, elle adopte un mode de fonctionnement cyclique, et permet à l'utilisateur d'intégrer tous types de données dans le lac, mais aussi diverses sortes d'analyse. De plus, pour la mise en œuvre concrète du lac de données qui répond au besoin métier, et plus particulièrement de son système de gestion des métadonnées, nous utilisons le métamodèle goldMEDAL, présenté dans le chapitre 3. Grâce à sa généricité, goldMEDAL peut aisément s'adapter aux problèmes auxquels nous faisons face. En outre, nous présentons HOUDAL (*public HOUsing DAta Lake*), une implémentation de lac de données basée sur notre architecture et adaptée à notre cas d'usage. Cette implémentation permet aux utilisateurs de créer et de gérer des métadonnées dans le lac de données et d'interagir avec les données stockées dans le lac.

La présentation de HOUDAL a fait l'objet d'un article publié dans une conférence internationale. E. Scholly, C. Favre, E. Ferey, S. Loudcher, "HOUDAL : a data lake implemented for public housing", *23rd International Conference on Enterprise Information Systems (ICEIS 2021), Prague, Czech Republic*, April 2021, 39-50 ; INSTICC, Setúbal, Portugal (Vol. 1) [Scholly et al., 2021a].

Ce chapitre est organisé comme suit. La section 4.2 présente le cas d'usage sur l'habitat social, ainsi que les problèmes rencontrés en utilisant les outils traditionnels pour stocker et analyser les données. Ensuite, nous présentons dans la section 4.3 notre architecture de lac de données, qui vise à ne pas compartimenter les données du lac, et qui se base sur quatre composants fonctionnant de manière cyclique. Dans la section 4.4, nous présentons le système de gestion de métadonnées du lac, en détaillant les concepts que nous utilisons ainsi que leur mise en œuvre au niveau physique. Suite à cela, la section 4.5 détaille HOUDAL, l'implémentation de notre proposition, en présentant les technologies employées pour son développement, ainsi que des perspectives d'évolution. Enfin, la section 4.6 conclut ce chapitre.

4.2 Cas d'usage

Pour expliquer au mieux les motivations et les enjeux des travaux de recherche présentés dans ce chapitre, nous commençons par présenter le cas d'usage réel qui a servi de fil directeur pour nous guider. Le cas d'usage, ancré dans la problématique métier de l'habitat social, est le projet « Etat Des Lieux » (EDL), initialement réalisé chez BIAL-X mais sans utiliser de lac de données. Plusieurs problèmes ont été rencontrés par les acteurs du projet, qui utilisaient des outils et méthodes traditionnels pour l'entreposage et l'exploitation des données. Ces problèmes rencontrés en l'absence d'un lac de données motivent justement son intérêt et son utilité.

Nous introduisons dans la section 4.2.1 les objectifs et les enjeux du projet EDL. Dans un second temps, la section 4.2.2 décrit les différents problèmes rencontrés lors de l'exploitation des données, sans bénéficier de la présence d'un lac de données.

4.2.1 Objectifs du projet EDL

Ce projet a été mené en collaboration avec un bailleur social avec qui BIAL-X travaille depuis maintenant plusieurs années. Par le passé, BIAL-X avait mis en place chez le bailleur tout un système d'information décisionnel, c'est-à-dire un entrepôt de données qui alimente divers tableaux de bords. Grâce aux informations tirées de ces derniers, le bailleur a pu avoir une meilleure connaissance et compréhension de ses données, et de nouveaux besoins d'analyse ont émergé. En particulier, le bailleur souhaitait pouvoir mieux gérer le départ des locataires dans l'optique de prédire si, d'une part, une visite pour l'état des lieux sortant est nécessaire, et d'autre part, si des travaux sont à prévoir dans le logement. L'objectif de ce projet était donc d'employer des méthodes de science des données pour fournir deux prédictions concernant le départ des locataires.

La première étude cherche à prédire si les équipements d'un logement risquent d'être dégradés lors du départ d'un locataire, et donc de prévoir ou non une visite de sortie. En effet, il est fréquent qu'un agent du bailleur se déplace pour effectuer la visite de l'état des lieux de sortie, mais si le locataire n'a rien endommagé, aucune compensation ne sera demandée au locataire sortant. Dans ce cas, il n'est pas nécessaire qu'une visite de sortie soit faite, et ainsi le bailleur économise les frais de déplacement de l'agent. L'objectif de cette prédiction est donc d'identifier à l'avance les locataires qui présentent une forte probabilité de rendre le logement avec des équipements dégradés, pour lesquels le bailleur pourrait demander une indemnité au locataire sortant, rendant ainsi nécessaire une visite pour l'état des lieux de sortie. Précisons qu'avec cette étude, nous nous focalisons sur les équipements endommagés ; les locataires restant très longtemps dans un logement et dont certains équipements peuvent présenter une usure « normale » ne sont pas concernés.

Une seconde étude a été menée pour prédire si, au départ du locataire, des travaux dans le logement seront nécessaires avant de pouvoir le relouer. L'objectif de cette prédiction est de réduire la période de vacance du logement et de pouvoir anticiper les travaux en les commandant avant le départ du locataire. En effet, si le constat est fait au départ du locataire que des travaux doivent être réalisés, le logement ne pourra pas être reloué immédiatement. Il faut donc compter la durée de la commande des travaux et le temps desdits travaux. Si les travaux sont commandés à l'avance, ils peuvent commencer dès le départ du locataire, ce qui réduit la période de vacance du logement et donc réduit les pertes du propriétaire, et contribue aussi à rendre le logement disponible plus rapidement pour de nouveaux locataires. Les travaux peuvent résulter d'une dégradation due au locataire (ce qui fait le lien avec la première étude), mais aussi et surtout de la vétusté du logement et donc du vieillissement normal de ses équipements : il n'y a ici pas de notion d'indemnité demandée au locataire.

Une fois les prédictions effectuées, les résultats ont été insérés dans un tableau de bord pour les fournir au bailleur. Avec ce tableau de bord, le bailleur peut facilement utiliser les résultats des deux prédictions et ainsi mieux visualiser les locataires susceptibles de rendre un logement avec des équipements endommagés, ou encore les logements pour lesquels des travaux seront à prévoir au départ du locataire.

4.2.2 Problèmes rencontrés

Lors du projet EDL, quatre principaux problèmes ont été rencontrés.

Multiples formats de données Le système d'information présent chez le bailleur et les outils associés ne permettaient pas de construire les modèles prédictifs demandés pour ce projet. Ainsi, pour le projet EDL, le bailleur a fourni des fichiers au format CSV qui sont extraits de son système d'information par des requêtes. Ensuite, les différents traitements de données effectués dans le cadre de ce projet ont généré de nouvelles données dans une grande variété de formats (par exemple, des fichiers au format .RData et .pkl pour les analyses R et Python ; des images .png générées avec des modèles prédictifs ; des rapports PowerBI au format .pbix ; et des documents de suivi de projet au format .ppt et .docx, entre autres). Finalement, les modèles prédictifs ont été créés puis enregistrés parmi ces données hétérogènes. Le stockage de données aux formats variés est toujours un défi. Les données sont ainsi dispersées dans différents environnements, ce qui peut rapidement compliquer la tâche des spécialistes des données. Avec un lac de données, ce problème n'existe plus car toutes les données se trouvent au même endroit, et restent accessibles dans la durée.

Plusieurs types d'analyse Ce projet a mobilisé plusieurs spécialistes des données : des consultants BI pour concevoir le tableau de bord de *reporting* et des *data scientists* pour construire des modèles prédictifs. Ces deux types de profils ne travaillent pas avec les mêmes outils et méthodes. L'intégration des résultats des prédictions dans le modèle de données de l'outil de *reporting* PowerBI [1] a été une tâche complexe : les besoins de modélisation pour des analyses décisionnelles ne sont pas les mêmes que pour des analyses de science des données. En outre, pour les seules analyses de science des données, le projet a été réalisé dans plusieurs environnements et technologies. Par exemple, le nettoyage des données qui servent à constituer le modèle en étoile à partir des données brutes a été effectué avec des scripts R. Les premiers essais pour la création des modèles prédictifs ont été menés via des scripts Python dans un *notebook* Jupyter. Ce dernier a par la suite été abandonné au profit de librairies Python permettant de gérer de manière automatique les flux de données pour la préparation des données puis l'entraînement des modèles. Combiné au fait que les données sont éparpillées, cela complique encore plus les choses puisque les analyses sont enregistrées à différents endroits et réalisées dans différents langages de programmation. En utilisant un lac de données, les traitements sont eux aussi centralisés, et il est ainsi bien plus facile d'interagir avec eux.

Lignage des données Le bailleur social a envoyé des données créées à partir de requêtes effectuées sur son entrepôt de données interne. Les extraits de données n'ont pas été générés en même temps ni par les mêmes personnes, si bien que les données fournies étaient parfois redondantes, et/ou non complètes. En conséquence, il existait plusieurs versions d'un fichier de données, certaines versions étant utilisées par certains scripts et

1. `https://powerbi.microsoft.com/`

pas d'autres. Avec des données et des analyses éparpillées, il n'était pas aisé de savoir quelles données sont utilisées, par qui, quoi et comment. Comme nous avions également très peu d'informations sur les redondances potentielles entre les données et les traitements, les informations se limitaient aux noms de fichiers ou aux commentaires dans le code. Ainsi, le lignage des données et le suivi clair des traitements pour les analyses n'a pas été effectué convenablement, ni enregistré.

Transfert de compétences Les trois problèmes mentionnés ci-dessus font qu'il est difficile, sans l'aide des spécialistes des données qui ont activement participé au projet, de comprendre le déroulement du travail, de savoir quelles données ont été exploitées, par qui, dans quel but, etc. Dans notre cas, il a fallu plusieurs réunions d'explication avec les différents acteurs du projet pour avoir une vision claire et précise de ce qui a été fait dans le projet, que ce soit de manière détaillée (fichier par fichier) ou de manière plus générale (objectifs). Si le projet avait été mené au sein d'un lac de données, la simple exploration des métadonnées aurait permis d'éviter ces réunions chronophages.

Ainsi, l'utilisation d'un lac de données permet en théorie de traiter ces problématiques difficiles à gérer avec les outils classiques. De ce fait, nous considérons comme pertinente l'utilisation d'un lac de données pour ce type de projet de Data Intelligence.

4.3 Architecture de lac de données

La section précédente nous a permis de poser le contexte métier et les motivations de nos travaux. Au regard des contraintes et des spécificités de notre cas d'usage, nous nous penchons maintenant sur la mise en place du lac de données, et plus particulièrement son architecture. Comme nous l'avons présenté dans le chapitre 2, la littérature scientifique propose plusieurs manières d'agencer l'architecture d'un lac de données, mais nous avons vu que toutes ces approches compartimentent les données du lac pour répondre à un besoin spécifique, ce que nous souhaitons éviter. C'est pourquoi nous proposons dans cette section notre propre architecture de lac de données, qui vise à éviter de compartimenter les données du lac, pour la rendre la plus flexible possible.

La section 4.3.1 confronte les architectures de lacs de données proposées dans la littérature scientifique au cas d'usage, et motive le besoin de proposer une approche différente. Nous continuons en présentant dans la section 4.3.2 les composants principaux qui constituent l'architecture du lac de données, ainsi que leur rôle. Ensuite, nous explicitons dans la section 4.3.3 le transit des données et des métadonnées à travers les différents composants, et comment l'utilisateur interagit avec le lac de données basé sur cette architecture. La section 4.3.4 conclut en présentant les avantages de notre approche par rapport aux autres architectures existantes.

4.3.1 Motivation pour une nouvelle architecture

La littérature scientifique propose plusieurs approches pour définir l'architecture d'un lac de données, avec entre autres des lacs de données organisés en zones [Ravat and Zhao,

2019b, LaPlante and Sharma, 2016] ou en bassins [Inmon, 2016, John and Misra, 2017]. D'autres auteurs préfèrent classer les architectures en trois catégories : fonctionnelles, en zones, et hybrides [Sawadogo and Darmont, 2021]. Toutefois, à travers un simple exemple illustratif, nous avons montré dans le chapitre 2 que ces propositions d'architectures compartimentent les données dès leur insertion, ce qui a un effet considérable sur l'utilisation du lac. Nous présentons ici pourquoi les architectures proposées dans la littérature s'adaptent mal au cas d'usage.

Le cas d'usage concerne principalement des données structurées, qui sont ensuite retravaillées à travers diverses opérations de traitement de données. Cependant, les données retravaillées finissent souvent par être stockées dans des formats peu courants, tels que des fichiers .RData ou .pkl. Bien que les données présentes dans ces fichiers soient tabulaires, ces formats non conventionnels induisent que les fichiers ne peuvent être exploités que par les outils adéquats. En effet, un fichier .RData (respectivement .pkl) doit être traité par un script R (respectivement Python). C'est pourquoi, pour la plupart des outils, ces fichiers sont considérés comme des données « non structurées » puisqu'ils ne peuvent pas être lus. Même s'il serait aisé de générer des fichiers .csv à partir des fichiers .RData ou .pkl, cela n'a pas été fait dans ce projet car ces formats sont optimisés pour les scripts R et Python, respectivement. Nous considérons donc qu'une approche basée sur la structure des données, avec la mise en place de bassins de données, n'est pas pertinente dans notre cas, car la variété des formats de données risque de créer des ambiguïtés.

En ce qui concerne les zones, les données sources dans leur format original (CSV) rempliraient la zone des données brutes, et le tableau de bord final serait dans la zone des données d'accès. Ces deux zones seraient peu remplies en comparaison avec la zone intermédiaire des données en traitement, qui contiendrait la plupart des données, avec des formats multiples. Ainsi, nous pensons que distinguer les données uniquement en fonction de leur degré de raffinement n'est pas la meilleure idée dans notre cas d'usage.

De ce fait, chaque architecture proposée a tendance à compartimenter les données du lac. En effet, en choisissant une architecture de lac basée sur le format des données (bassins) ou leur degré de raffinement (zones), les données sont immédiatement séparées dans des espaces disjoints à leur insertion, en fonction de leurs propriétés intrinsèques. Nous considérons que ces approches sont contraires au principe de fonctionnement *schema-on-read* d'un lac de données, où les données sont enregistrées sans schéma prédéfini. Au contraire, nous préférons étiqueter les données lors de leur ingestion dans le lac, et les catégoriser selon leur format et leur niveau de maturité, entre autres. Par la suite, l'utilisateur peut alors, à la demande, rassembler les données du lac en bassins ou en zones à l'aide des catégorisations qui sont enregistrées dans les métadonnées. Pour ces raisons, nous préférons ne pas nous baser sur une architecture existante, mais plutôt lister les composants qui nous paraissent importants pour le bon fonctionnement du lac.

4.3.2 Composants nécessaires

La figure 4.1 illustre notre proposition d'architecture à travers un schéma. Le lac de données est constitué des parties situées dans l'encadré en pointillé. Nous distinguons les quatre composants de l'architecture, que nous présentons en détail ci-dessous : l'inges-

tion (en orange) par laquelle les données sources passent pour créer les métadonnées ; le stockage des données et le système de gestion de métadonnées (en vert) ; et l'exploitation des données (en violet).

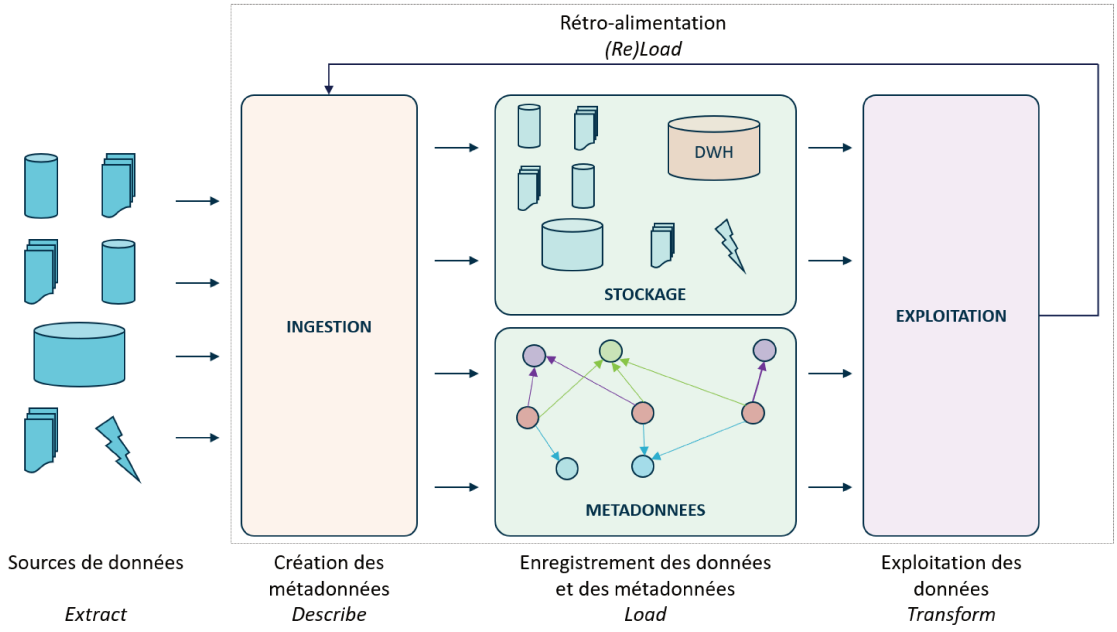

FIGURE 4.1 – Proposition d'architecture de lac de données

Le premier composant du lac de données est la couche de **stockage des données**, qui peut contenir tout type de données. Rappelons que le lac de données est avant tout considéré comme un système pouvant entreposer des données de tous types et dans leur format natif. Le **système de métadonnées** constitue le second composant majeur de toute architecture de lac de données. Les métadonnées servent à décrire les données stockées dans le lac, et aide l'utilisateur à naviguer à l'intérieur du lac pour pouvoir s'y retrouver. Rappelons que sans système de métadonnées efficace, il est difficile voire impossible de s'y retrouver, et le lac de données inutilisable devient alors un marécage de données.

Nous pensons qu'il est pertinent d'ajouter à l'architecture du lac de données un composant dédié à l'**ingestion** des données. En effet, la création des métadonnées est un processus primordial pour le bon fonctionnement du lac, mais si l'utilisateur doit toutes les saisir manuellement, même le plus efficace des systèmes de métadonnées peut devenir très fastidieux à utiliser. C'est là que cette couche d'ingestion des données entre en jeu : son objectif est de faciliter la création de métadonnées en générant automatiquement autant de métadonnées que possible, ce qui permet aux utilisateurs de valider ou d'invalider rapidement les métadonnées. Cette couche est également importante pour éviter que l'utilisateur ne dépose des données dans le lac sans saisir de métadonnées, ce qui aurait pour conséquence de transformer peu à peu le lac en marécage de données.

Les données du lac ne sont pas seulement destinées à être stockées sans être exploitées.

C'est pourquoi nous introduisons un quatrième composant qui est l'**exploitation** du lac de données, au sein duquel tous types d'utilisation et d'analyse de données peuvent être envisagés. L'idée est de laisser à l'utilisateur la possibilité de mener n'importe quel type d'exploitation de données dont il pourrait avoir besoin. À ce titre, il est envisageable de considérer que ce composant soit avant tout une mise à disposition des données via un service dédié, pour que l'utilisateur puisse y connecter ses outils d'exploitation favoris. Par exemple, l'utilisateur pourrait souhaiter analyser les données structurées à l'aide d'un outil de *reporting*, ou encore traiter les données dans le but d'effectuer une prédiction avec une librairie Python dans un *notebook* adapté.

4.3.3 Fonctionnement de l'architecture

Pour résumer le fonctionnement de l'architecture, les données sont extraites des sources et passent par la couche d'ingestion. Là, les métadonnées sont créées puis enregistrées dans le système de métadonnées, tandis que les données sont déposées dans la couche de stockage. Ensuite, l'utilisateur peut parcourir les métadonnées pour s'informer sur les données présentes dans le lac, et éventuellement sélectionner les jeux de données qui peuvent lui servir pour un besoin. Il peut ensuite mener tous types d'analyses sur les données sélectionnées.

Dans certains cas, il se peut qu'une exploitation de données du lac génère de nouvelles données, et que l'utilisateur estime pertinent d'enregistrer ces nouvelles données dans le lac. Alors, les nouvelles données sont envoyées dans la couche d'ingestion afin de créer les métadonnées adéquates, puis sont enregistrées dans la couche de stockage. Nous désignons ce mode de fonctionnement comme la « rétroalimentation » du lac de données.

La propriété *schema-on-read* d'un lac de données implique que le schéma de données est spécifié lors de la requête. C'est pourquoi les lacs de données fonctionnent en mode *Extract - Load - Transform* (ELT), qui s'oppose au mode *Extract - Transform - Load* (ETL) typique des entrepôts de données, où les données sont transformées avant d'être enregistrées. Dans le lac de données, les données ne sont pas transformées avant d'être stockées, mais seulement au moment de l'interrogation. Dans notre cas, pour mieux décrire notre architecture, nous préférons dire que le lac de données fonctionne en EDLT(L) :

- *Extract* : les données sont extraites des sources ;
- *Describe* : les données à insérer sont décrites par des métadonnées ;
- *Load* : les données sont enregistrées dans le lac de données, sans aucune modification ;
- *Transform* : l'utilisateur transforme les données interrogées selon ses besoins d'exploitation ;
- *(Re)Load* : si l'utilisateur souhaite sauvegarder le résultat de son exploitation, il peut « rétroalimenter » le lac et sauvegarder les données transformées, accompagnées de métadonnées adéquates.

4.3.4 Discussion : avantages de notre architecture

Avec le composant d'ingestion des données et en ajoutant l'étape *Describe*, nous insistons sur l'importance de l'étape de création des métadonnées. En effet, nous considérons qu'elles doivent être générées *avant* de stocker les données dans le lac de données. Sauvegarder des données dans le lac sans capturer les métadonnées associées représente un risque important de perdre leur trace et de transformer rapidement le lac en un marécage inutilisable. Nous pouvons ainsi, à la manière de la propriété *schema-on-read* des lacs de données, ajouter une nouvelle propriété *describe-on-write*, c'est-à-dire qu'il est primordial de correctement décrire les données avant de les enregistrer dans le lac.

De plus, avec l'étape de *(Re)Load*, nous insistons sur le fait que le lac doit être la pièce centrale de toute la chaîne d'analyse des données. Bien sûr, les données sont d'abord extraites de sources extérieures au lac ; cependant, lorsque l'utilisateur retravaille les données du lac pour un besoin donné, les résultats doivent être stockés dans le lac. Ainsi, pour ces données retravaillées, le cycle « redémarre » à l'étape *Describe*, c'est-à-dire que les nouvelles données passent par la couche d'ingestion. À notre connaissance, notre architecture est la seule proposant ce fonctionnement cyclique, qui permet au lac de s'auto-alimenter. Ceci présente le lac de données comme un système vivant, avec lequel l'utilisateur interagit et travaille, et non comme un simple dépôt de données issues de sources externes.

Ajoutons aussi que le composant d'exploitation des données peut être considéré à la fois comme étant situé à l'intérieur du lac de données, mais aussi en dehors. En effet, nous souhaitons que l'utilisateur soit en mesure d'interroger les données du lac à l'aide d'outils qui ne sont pas forcément inclus dans le lac. Par exemple, l'utilisateur pourrait interroger les données du lac à l'aide d'un script qui interroge un service dédié mis à disposition par le composant d'exploitation. L'important ici est que, si des données sont générées grâce à cette exploitation, elles rétroalimentent alors le lac en passant par une phase d'ingestion. Le cas échéant, le script ayant servi pour transformer les données peut aussi être enregistré dans le lac si l'utilisateur estime que c'est nécessaire.

4.4 Système de gestion de métadonnées

Dans la section 4.2, nous avons présenté comment le cas d'usage génère des problèmes lors de l'utilisation des outils traditionnels pour l'entreposage et l'analyse de données. Nous avons également montré que, d'un point de vue théorique, l'utilisation d'un lac de données serait une bonne solution pour résoudre ces problèmes. Afin de valider notre proposition, nous avons essayé de répliquer dans HOUDAL le projet EDL, qui avait été initialement mené sans utiliser de lac de données. Nous examinons ici le système de métadonnées de HOUDAL lorsqu'il est appliqué à ce cas d'usage, et comment il aide les spécialistes des données.

Nous commençons par présenter dans la section 4.4.1 comment se déclinent, dans le cadre de notre cas d'usage métier, les différents éléments composant notre typologie de métadonnées. Ensuite, la section 4.4.2 nous permet de détailler comment nous instancions

le métamodèle conceptuel de goldMEDAL en un modèle conceptuel, que nous traduisons ensuite au niveau logique. Nous terminons par la section 4.4.3 dans laquelle nous déclinons le modèle logique en un modèle physique, basé sur le système de gestion de base de données graphe Neo4j.

4.4.1 Typologie des métadonnées dédiées au cas d'usage

Parmi nos propositions du chapitre 3, nous avons introduit une typologie de métadonnées. Elle repose sur la notion d'objet qui désigne un ensemble homogène de données, et rassemble les métadonnées en trois catégories : intra-objet, inter-objets et globales. Nous détaillons ici la forme que prennent ces métadonnées en appliquant la typologie au cas d'usage métier.

4.4.1.1 Métadonnées intra-objet

Rappelons que les métadonnées intra existent sous plusieurs types : les propriétés, les métadonnées sémantiques, les résumés et prévisualisations, ainsi que les versions et représentations.

Les propriétés d'un objet prennent la forme d'attributs que nous affectons à chaque objet. Nous en listons cinq.

- Intitulé du fichier (exemple : `logements_2021.csv`)
- Taille, en octets (exemple : 3.78 Mo)
- Auteur, i.e. l'utilisateur ayant inséré les données dans le lac (exemple : ESY)
- Date d'insertion (exemple : 28/09/2021)
- Format (exemple : CSV)

En ce qui concerne les métadonnées sémantiques, l'utilisateur saisit une description textuelle à l'insertion des données dans le lac. En reprenant l'exemple du fichier `logements.csv`, l'utilisateur peut saisir la description « Informations de 2021 sur les logements du bailleur ». Cette description est enregistrée en tant qu'attribut, avec les propriétés.

Les objets peuvent aussi avoir des versions, c'est-à-dire de nouvelles données mises à jour, ou des représentations, lorsque l'utilisateur retravaille les données pour un besoin spécifique. Ce cas de figure arrive plusieurs fois dans notre cas d'usage, car les bailleurs ont envoyé des nouvelles versions des données, et les spécialistes des données ont dû retravailler les données avant de construire des modèles prédictifs.

Pour ce cas d'usage, nous ne disposons pas de résumé ou prévisualisation des données.

4.4.1.2 Métadonnées inter-objets

Dans la typologie, trois types de métadonnées inter existent : les regroupements d'objets, les liaisons de similarité et les relations de parenté.

Les regroupements d'objets permettent de rassembler des objets ayant des propriétés communes. Nous pouvons par exemple imaginer un regroupement d'objets sur le secteur métier des données, où chaque collection d'objets correspond à un secteur (locatif, impayés, finance, RH, etc.).

Les relations de parenté permettent de gérer le cas de figure où des données de différents objets sont croisées pour former un nouvel objet. Nous rencontrons ceci dans notre cas d'usage, en particulier lors de la jointure de différents fichiers pour constituer la matrice qui servira à l'entraînement des modèles prédictifs. Les relations de parenté servent ainsi à montrer que les données, initialement disjointes, sont croisées jusqu'à obtenir les deux résultats attendus du projet, à savoir les deux prédictions.

Nous n'avons en revanche pas de liaison de similarité. Toutefois, nous en proposons dans le cadre d'une nouvelle problématique métier. Ces travaux feront l'objet du chapitre 5.

4.4.1.3 Métadonnées globales

Pour rappel, trois types de métadonnées globales existent : les ressources sémantiques, les *logs* et les index (et index inversés).

Dans le cadre de ce cas d'usage, nous n'avons pas de ressource sémantique. Nous avons en effet suffisamment de catégorisations métier, gérées avec les regroupements d'objets, pour rassembler les données selon des critères prédéfinis. De même, nous n'avions pas besoin de disposer d'index ou d'index inversé.

Nous ne disposons pas non plus de *logs*, mais ils pourraient en revanche nous servir pour mieux suivre les travaux des différents acteurs du projet. Nous pourrions en effet savoir quel utilisateur a consulté quelles données, pour tenter de mieux retracer les évènements. Malheureusement, nous n'étions pas en mesure de créer ces journaux d'évènements en recréant le projet dans le lac de données. Il serait toutefois pertinent d'y inclure la génération automatique de *logs* pour des travaux futurs.

4.4.2 Instanciation des métamodèles conceptuel et logique

Le système de gestion de métadonnées de HOUDAL se base sur le métamodèle de métadonnées goldMEDAL. Nous étudions ici comment nous pouvons instancier le métamodèle conceptuel de goldMEDAL en un modèle conceptuel. Nous traduisons ensuite chaque concept du modèle au niveau logique à l'aide de la théorie des graphes.

4.4.2.1 Entité de données

Les différents fichiers de données qui peuplent le lac de données sont modélisés comme des entités de données. Il peut s'agir de fichiers de données brutes envoyés par les bailleurs (souvent des fichiers CSV) ou de données retravaillées, parfois stockées dans différents formats tels que .pkl ou .RData, pour les analyses avec Python ou R, respectivement.

Au niveau logique, une entité de données notée e est représentée par un nœud porteur d'attributs, noté n.

4.4.2.2 Groupement

Un groupement est un ensemble de groupes, et les groupes sont utilisés pour catégoriser les entités de données. Avec HOUDAL, l'utilisateur peut créer autant de groupements que nécessaire, et plusieurs groupes pour chaque groupement. Les entités de données peuvent être liées à zéro, un ou plusieurs groupes pour chaque groupement, en fonction de la nature du groupement (partition ou non). Par exemple, les différentes zones désignant le degré de raffinement de données peut constituer un groupement, où chaque groupe représente une zone. Une catégorisation métier peut être un second exemple de groupement.

Au niveau logique, un groupe est modélisé par une hyperarête (c'est-à-dire une arête pouvant lier plus de deux nœuds), un groupement est donc un ensemble d'hyperarêtes.

Exemple 9 *Soit* $G_1 = \{\Gamma_{11}, \Gamma_{12}, \Gamma_{13}\}$ *un groupement représentant les différentes zones de données pour désigner leur degré de raffinement. Les groupes* Γ_{11}, Γ_{12} *et* Γ_{13} *désignent respectivement la zone de données brutes, la zone de traitement et la zone d'accès. En outre, soit* $G_2 = \{\Gamma_{21}, \Gamma_{22}\}$ *un groupement visant à catégoriser les entités de données selon leur signification sémantique. Ici,* Γ_{21} *et* Γ_{22} *désignent les données portant sur les logements et les locataires, respectivement.*

Rendons cet exemple plus concret avec des instances. Soit trois entités de données $\{e_1, e_2, e_3\}$. e_1 *et* e_2 *sont des données brutes, tandis que* e_3 *sont des données retravaillées. Ainsi,* $\Gamma_{11} = \{e_1, e_2\}$ *et* $\Gamma_{12} = \{e_3\}$. *En outre, les données de* e_2 *portent sur les logements, alors que* e_1 *et* e_3 *sont des fichiers concernant les locataires. Alors,* $\Gamma_{21} = \{e_2\}$ *et* $\Gamma_{22} = \{e_1, e_3\}$.

Au niveau logique, l'ensemble d'hyperarêtes représentant les zones est noté $H_1 = \{\theta_{11}, \theta_{12}, \theta_{13}\}$, *et celui désignant la sémantique est données est* $H_2 = \{\theta_{21}, \theta_{22}\}$.

Représentons le niveau logique de l'exemple 9 avec la figure 4.2. Les nœuds sont colorés en orange, et les groupes de H_1 et H_2 sont colorés en violet et bleu, respectivement. Nous constatons que n_1 et n_2 appartiennent au groupe de données brutes θ_{11}, tandis que n_3 est dans le groupe représentant la zone de traitement θ_{12}. De plus, n_2 est dans le groupe « logements » θ_{21}, et n_1 et n_3 font partie du groupe « locataires » θ_{22}. Notons que le groupe représentant la zone d'accès θ_{13} n'est pas représenté sur cette figure.

4.4.2.3 Lien

Dans goldMEDAL, il existe deux types de liens : les liens reliant des entités de données entre elles et les liens hiérarchiques entre groupes d'un groupement. Concernant le premier type de lien, nous n'en avons pas besoin pour modéliser les éléments du cas d'usage. Toutefois, nous en aurons besoin dans une deuxième problématique métier, et ces travaux seront présentés en détail dans le chapitre 5.

Les liens hiérarchiques entre groupes peuvent nous aider à constituer des catégorisations métier à plusieurs niveaux, et donc plus complexes. Par exemple, nous disposons d'un groupement sur le format des données, ou chaque groupe désigne un format (CSV,

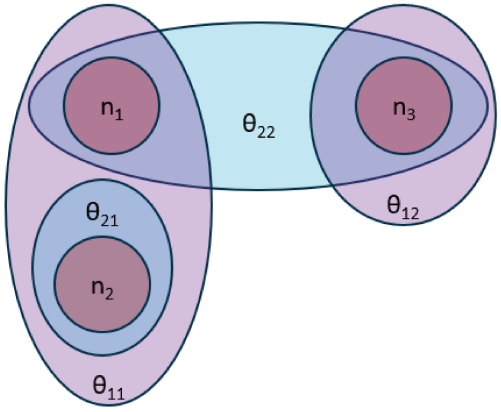

FIGURE 4.2 – Groupement de goldMEDAL appliqué au cas d'usage

XLS, JPG, PNG, etc.). Nous pouvons alors créer un second groupement « bassin », composé de trois groupes (structuré, semi-structuré, non structuré), et relier entre eux les groupes de ces deux groupements par un lien hiérarchique. Ici, les groupes CSV et XLS sont reliés au groupe structuré, tandis que les groupes JPG et PNG sont connectés au groupe non structuré.

Ce type de lien se traduit au niveau logique par une arête orientée reliant deux hyperarêtes.

Exemple 10 *Soit* $G_2 = \{\Gamma_{CSV}, \Gamma_{XLS}, \Gamma_{JPG}, \Gamma_{PNG}\}$ *l'ensemble des formats, chaque groupe désignant un format (CSV, XLS, JPG et PNG, respectivement). Soit* $G_3 = \{\Gamma_{31}, \Gamma_{32}\}$ *l'ensemble des bassins de données, avec* Γ_{31} *représentant l'ensemble des données structurées, et* Γ_{32} *l'ensemble des données semi-structurées ou non structurées. Le lien hiérarchique* l_1 *relie les formats aux bassins correspondants :* $\Gamma_{CSV} \underset{l_1}{\longrightarrow} \Gamma_{31}$, $\Gamma_{XLS} \underset{l_1}{\longrightarrow} \Gamma_{31}$, $\Gamma_{JPG} \underset{l_1}{\longrightarrow} \Gamma_{32}$ *et* $\Gamma_{PNG} \underset{l_1}{\longrightarrow} \Gamma_{32}$. *Inversement,* $\Gamma_{31} \underset{l_1^{-1}}{\longrightarrow} \{\Gamma_{CSV}, \Gamma_{XLS}\}$ *et* $\Gamma_{32} \underset{l_1^{-1}}{\longrightarrow} \{\Gamma_{JPG}, \Gamma_{PNG}\}$.

Traduisons maintenant cela au niveau logique. L'arête a_1 *matérialise le lien hiérarchique entre* H_2 *et* H_3 *:* $\theta_{CSV} \underset{a_1}{\longrightarrow} \theta_{31}$, $\theta_{XLS} \underset{a_1}{\longrightarrow} \theta_{31}$, $\theta_{JPG} \underset{a_1}{\longrightarrow} \theta_{32}$ *et* $\theta_{PNG} \underset{a_1}{\longrightarrow} \theta_{32}$. *Aussi, nous avons* $\theta_{21} \underset{a_1^{-1}}{\longrightarrow} \{\theta_{CSV}, \theta_{XLS}\}$ *et* $\theta_{32} \underset{a_1^{-1}}{\longrightarrow} \{\theta_{JPG}, \theta_{PNG}\}$.

La figure 4.3 illustre l'exemple 10. Le lien a_1 connecte les groupes de H_2 (θ_{CSV}, θ_{JPG}, etc.), colorés en vert, aux groupes de H_3, qui sont eux colorés en gris.

4.4.2.4 Processus

Les processus reflètent les modifications que les données peuvent subir, et sont utilisés pour suivre le lignage des données. Il peut s'agir, par exemple, d'un script de transformation de données brutes pour les nettoyer, en vue d'une future analyse.

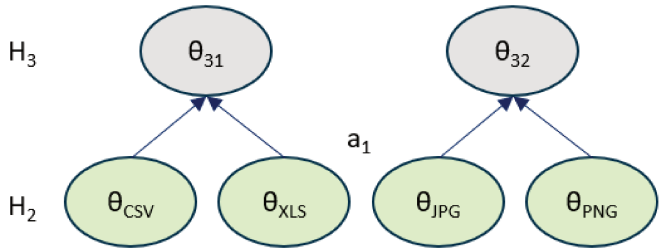

FIGURE 4.3 – Lien hiérarchique de goldMEDAL appliqué au cas d'usage

Au niveau logique, un processus est matérialisé par une hyperarête orientée reliant des nœuds.

Exemple 11 *Soit e_4, e_5 et e_6 trois entités de données modélisant des données brutes, et e_7, e_8 et e_9 trois autres entités de données représentant des données nettoyées. Le processus $P_1 = \{I_1, O_1\}$ matérialise le script de nettoyage des entités de données brutes, avec $I_1 = \{e_4, e_5, e_6\}$ l'ensemble des entités de données en entrée du processus P_1, et $O_1 = \{e_7, e_8, e_9\}$ l'ensemble des entités de données en sortie du processus P_1.*

Au niveau logique, nous notons $\Pi_1 = \{\Upsilon_1, \Omega_1\}$, avec $\Upsilon_1 = \{n_4, n_5, n_6\}$ l'ensemble des nœuds d'entrée de Π_1, et $\Omega_1 = \{n_7, n_8, n_9\}$ l'ensemble des nœuds de sortie de Π_1.

Nous décrivons l'exemple 11 de manière visuelle à travers la figure 4.4. Les nœuds sont à nouveau colorés en rouge, et l'hyperarête orientée Π_1 est colorée en jaune. Les trois nœuds n_4, n_5, n_6 sont en entrée du processus Π_1, c'est-à-dire dans Υ_1, tandis que les nœuds n_7, n_8, n_9 sont dans Ω_1, i.e. en sortie du processus.

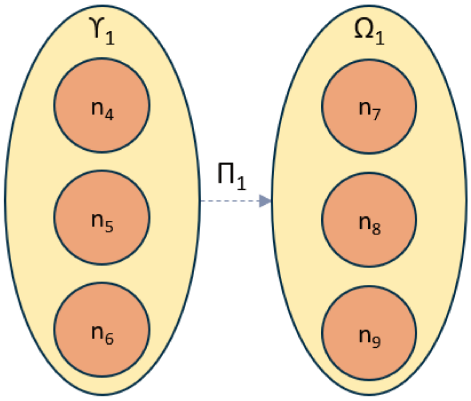

FIGURE 4.4 – Processus de goldMEDAL appliqué au cas d'usage

4.4.3 Modèle physique du cas d'usage

Pour son système de métadonnées, HOUDAL prend appui sur l'instanciation du modèle logique de goldMEDAL présentée précédemment, que nous traduisons en un modèle

physique. Celui-ci est stocké au sein du SGBD graphe Neo4j, qui est particulièrement adapté pour représenter les relations. Nous présentons comment les différentes métadonnées sont enregistrées dans Neo4j.

4.4.3.1 Entité de données

Dans Neo4j, chaque entité de données est un nœud étiqueté `:ENTITY` et les propriétés de l'entité, comme le nom du fichier ou la description, sont stockées dans les attributs du nœud. Ces informations décrivent les données de manière individuelle.

Nous donnons dans la figure 4.5 un exemple d'entité de données enregistrée dans Neoj sous la forme d'un nœud possédant des attributs. Nous constatons que l'identifiant du nœud est 192, qu'il possède l'étiquette (ou label) `:ENTITY`, mais aussi qu'il dispose de plusieurs propriétés : nom du fichier, taille en octets, date de création, auteur, description textuelle, nombre de lignes, nombre d'attributs, et délimiteur. Précisons que dans Neo4j, un nœud peut posséder plusieurs étiquettes.

Cette entité de données désigne un fichier CSV portant sur des enquêtes effectuées par le bailleur social auprès de ses locataires pour mettre à jour les informations dont il dispose. Précisons que le délimiteur, le nombre de lignes et le nombre de colonnes sont des attributs spécifiques pour les données au format CSV, et sont utilisés plus spécifiquement dans les travaux du chapitre 5.


```
{
  "identity": 192,
  "labels": [
    "ENTITY"
  ],
  "properties": {
"date": "2020-07-03T08:55:22.206Z",
"file": "2019-07-17 EDL_3 Frais Non-Rep Enquetes.csv",
"size": 749268,
"delimiter": ";",
"author": "esy",
"description": "Survey about the tenant's current situation",
"nb_rows": 6404,
"nb_cols": 17
  }
}
```


FIGURE 4.5 – Exemple d'entité de données dans HOUDAL (Neo4j)

4.4.3.2 Groupement

Dans le modèle physique, les groupements sont modélisés par des nœuds portant une étiquette `:GROUPING`. Les groupes sont également des nœuds, portant à la fois l'étiquette

`:GROUP` et le nom du groupement comme deuxième étiquette, afin de faciliter l'interrogation. Un nœud d'entité de données peut être lié à plusieurs nœuds de groupe.

Un nœud d'entité de données (resp. nœud de groupe) est lié à un nœud de groupe (resp. nœud de groupement) par une arête étiquetée avec le nom du groupement (resp. `:GROUPING`).

La figure 4.6 donne un exemple de groupements stockés dans Neo4j. Les nœuds d'entités de données sont colorés en rouge. Trois groupements sont visibles sur la figure : un groupement de zone, un groupement de format et un groupement de granularité. Chaque groupement a ses nœuds de groupe, colorés en vert, violet et bleu, respectivement. Les nœuds d'entité de données sont reliés aux nœuds de groupement par une arête. Par exemple, nous pouvons voir que le nœud d'entité de données dont l'identifiant est 192 (à gauche) est un fichier CSV, brut, et son niveau de granularité est *Tenant*, ce qui signifie que chaque observation du fichier de données désigne un locataire. Notons que dans Neo4j, les groupements sont aussi modélisés comme des nœuds, mais ne sont pas représentés dans cette figure.

Dans le cas d'usage, les groupes et groupements permettent à l'utilisateur de retrouver rapidement les entités de données qu'il recherche. Par exemple, il est aisé de retrouver toutes les données brutes, ou encore celles au format CSV.

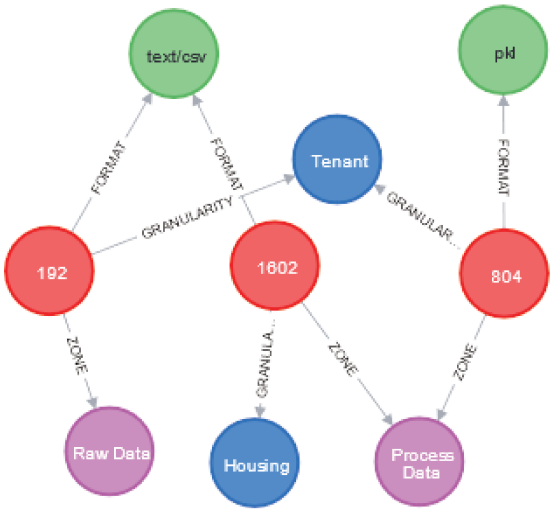

FIGURE 4.6 – Exemple de groupement dans HOUDAL (Neo4j)

4.4.3.3 Lien

Pour représenter un lien hiérarchique entre groupes dans Neo4j, nous utilisons plusieurs arcs. Premièrement, un groupe du groupement « inférieur » (c'est-à-dire celui de niveau de granularité le plus fin) est lié au groupe du groupement « supérieur » (i.e. celui au niveau de granularité plus élevé) par une arête orientée et étiquetée `:HIERARCHY`. De

même, nous relions le nœud du groupement « inférieur » à celui du groupement « supérieur » par une arête orientée et étiquetée `:HIERARCHY`, à nouveau.

La figure 4.7 montre comment les liens hiérarchiques sont enregistrés dans Neo4j. Les nœuds colorés en bleu sont les groupes du groupement « Granularité », lui-même modélisé par un nœud beige au bas de la figure. Nous constatons que ces groupes sont reliés par des arêtes étiquetées `:HIERARCHY` aux nœuds (colorés en rose) du second groupement « Sémantique ». Ce dernier est aussi représenté par un nœud beige, et le nœud du groupement de granularité est relié au nœud du groupement sémantique par une autre arête étiquetée de la même façon.

Sur la partie gauche de la figure, précisons que le groupe *Tenant* du groupement « Sémantique » (en rose) vise à rassembler les entités de données dont le contenu porte sur les locataires, tandis que le groupe aussi appelé *Tenant* mais du groupement « Granularité » (en bleu) désigne les entités de données dont chaque observation désigne un locataire. De même, sur la partie droite, le groupe *Housing* du groupement « Sémantique » (en rose) rassemble les entités de données portant sur les logements, alors que le groupe *Housing* du groupement « Granularité » (en bleu) catégorise les entités de données pour lesquelles une observation correspond à un logement.

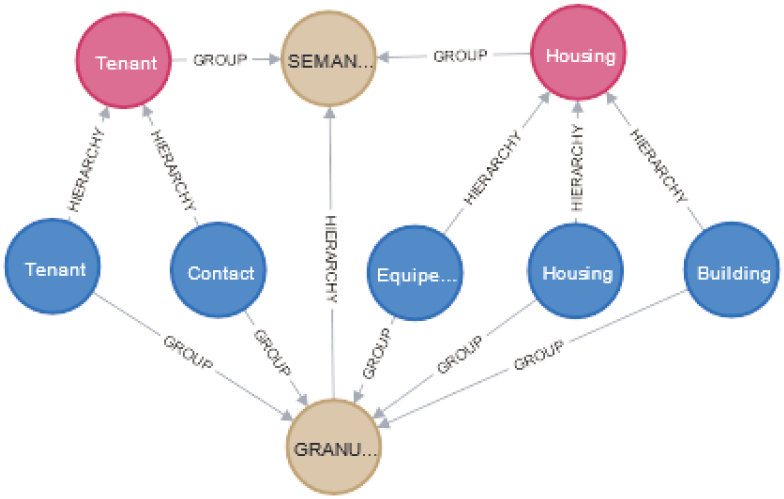

FIGURE 4.7 – Exemple de lien hiérarchique dans HOUDAL (Neo4j)

4.4.3.4 Processus

Dans Neo4j, un processus est aussi modélisé par un nœud, étiqueté `:PROCESS`. Les entités de données peuvent être en entrée du processus (par exemple, les données brutes envoyées par le propriétaire), ou en sortie (données nettoyées). Si une entité de données est en entrée d'un processus, il existe une arête étiquetée `:PROCESS_IN` entre le nœud de l'entité et le nœud du processus. Inversement, si le processus génère une nouvelle entité de données, une arête étiquetée `:PROCESS_OUT` est créée entre le nœud du processus et le

nœud de l'entité.

La figure 4.8 présente un exemple de processus enregistré dans Neo4j. Les nœuds d'entités de données sont à nouveau colorés en rouge, et le nœud du processus est coloré en jaune. Nous pouvons voir que trois nœuds d'entité de données sont en entrée du processus (à gauche), et trois nœuds d'entité de données constituent la sortie du processus (à droite), ce qui signifie qu'ils sont générés par le processus.

Dans le cas d'usage, ce processus désigne un traitement qui prend en entrée des brutes pour les retravailler et fournir des données nettoyées. C'est un résultat intermédiaire : d'autres traitements utilisent les données nettoyées par la suite.

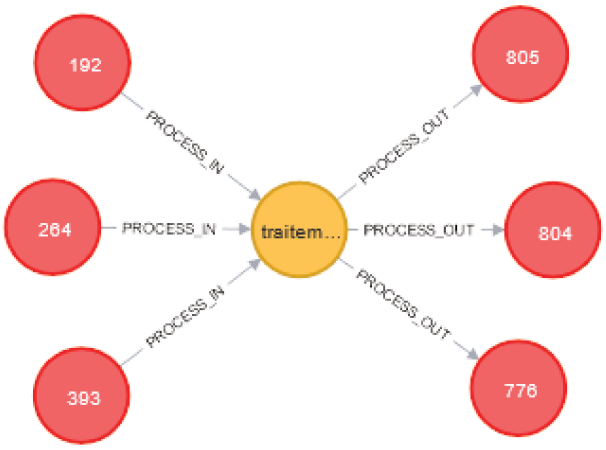

FIGURE 4.8 – Exemple de processus dans HOUDAL (Neo4j)

4.5 Implémentation du lac de données

Les sections précédentes nous ont permis de détailler l'architecture du lac de données, puis de présenter de manière exhaustive le système de gestion de métadonnées du lac, pour illustrer comment nous instancions le métamodèle goldMEDAL pour mettre en œuvre le modèle physique. Nous souhaitons désormais exposer l'implémentation de HOUDAL, en présentant les différents services qui le composent et étudier leurs interactions. L'objectif de cette implémentation est de répondre aux problèmes du cas d'usage, mais aussi de valider notre proposition d'architecture.

La section 4.5.1 détaille les options que nous avions à notre disposition pour l'implémentation, et présente les choix technologiques effectués. Ensuite, nous présentons dans la section 4.5.2 les différents services de HOUDAL, la manière dont ils communiquent entre eux, ainsi que leurs spécifications techniques. La section 4.5.3 nous permet de discuter des forces et faiblesses de notre outil, et de donner notamment des axes d'amélioration. Enfin, la section 4.5.4 montre comment HOUDAL contribue à pallier les problèmes rencontrés lors du projet EDL.

4.5.1 Choix des technologies

Bien que le concept de lac de données soit encore relativement récent et qu'il n'y ait pas encore de consensus sur l'architecture qu'un lac de données devrait avoir, il existe déjà plusieurs outils et technologies disponibles pour mettre en œuvre ce type de solution. Nous avons d'ailleurs présenté dans le chapitre 2 un éventail de possibilités pour la mise en place technique d'un lac de données. Nous nous appuyons sur cette étude pour identifier les outils qui peuvent être utilisés pour répondre aux exigences du cas d'usage, et qui permettent d'implémenter l'architecture de lac proposée. Ainsi, nous présentons les diverses options considérées en expliquant et justifiant les choix effectués. Nous avons exploré quatre possibilités : les outils « tout-en-un », les *clouds* propriétaires, l'écosystème Hadoop ou un assemblage *ad hoc* de technologies ouvertes.

4.5.1.1 Outil « tout-en-un »

La première option consiste à utiliser un outil tout-en-un comme Apache Kylo [2]. Celui-ci est défini comme « une plateforme logicielle de gestion de lac de données en libre-service pour l'acquisition et la préparation de données avec gestion intégrée des métadonnées ». Kylo gère à la fois le stockage et l'analyse des données, ainsi que le système de métadonnées. Cependant, cette option a été abandonnée pour trois raisons principales. Tout d'abord, le système de métadonnées proposé par Kylo n'est pas assez exhaustif pour nous permettre de mettre en œuvre notre système de métadonnées basé sur goldMEDAL. De plus, Kylo est principalement orienté vers la gestion de données structurées. Bien que nous ayons principalement des données sources au format .csv dans notre cas d'usage, cela devient problématique lorsque nous traitons des fichiers de différents formats. La dernière raison concerne l'analyse des données : la partie préparation des données de l'outil est certes intéressante, surtout dans notre cas, mais en ce qui concerne les analyses plus avancées (*reporting* ou prédictions, par exemple), nous aurions dû les gérer en dehors de Kylo. Cela aurait eu plusieurs conséquences néfastes sur l'utilisabilité du lac, comme la gestion des métadonnées en dehors de Kylo.

4.5.1.2 *Clouds* propriétaires

La deuxième piste envisagée consiste à utiliser des *clouds* (ou nuages) propriétaires, tels que Amazon Web Services (AWS) ou Microsoft Azure. Tous deux ont pour principal avantage d'offrir une large gamme de services, permettant d'ingérer tous types de données, de gérer les métadonnées et de réaliser tous types d'analyses de données. Il est donc possible de rester dans le même écosystème pour tous les besoins d'analyse, et il n'y a plus à se soucier du déploiement du serveur adéquat.

Cependant, le fait de tout avoir dans le *clouds*, et des services pour tout, s'avère être un inconvénient. Tout d'abord, si des développements ont déjà été réalisés, il faut les migrer vers les services appropriés au sein du *cloud*, ce qui peut se révéler très chronophage. Par ailleurs, nous avons constaté que ces *clouds* propriétaires ne sont pas flexibles :

2. https ://kylo.io/

en particulier, les services de gestion des métadonnées manquaient d'exhaustivité sur certains aspects, et nous aurions aimé pouvoir y apporter quelques modifications pour mieux correspondre à nos attentes, mais cela s'est avéré impossible. Enfin, les *clouds* propriétaires sont payants, ce qui nous a rebuté pour une première preuve de concept.

Cependant, ils pourraient être reconsidérés pour des travaux futurs sur des cas d'usage plus larges ou avec des besoins différents.

4.5.1.3 Ecosystème Hadoop

Une troisième option explorée consiste à utiliser un écosystème issu des technologies ouvertes pour la gestion des mégadonnées, comme Apache Hadoop. Comme les *clouds* propriétaires, Hadoop est accompagné de son propre écosystème, et de nombreux services sont disponibles pour ingérer des données, les stocker (notamment avec le *Hadoop Distributed File System*, HDFS), gérer le système de métadonnées et exécuter différents types d'analyse de données. En particulier, le système de métadonnées Apache Atlas est très intéressant, et semble être suffisamment complet pour permettre d'implémenter notre système de métadonnées.

Néanmoins, suite à des tests préliminaires, cette solution n'a finalement pas été retenue. D'une part, HDFS est adapté aux gros fichiers, mais pas aux petits fichiers, que nous avons en grande quantité dans notre cas d'usage. D'autre part, l'installation de tout un écosystème Apache Hadoop est un processus très long et complexe, et nécessite des compétences spécifiques, en plus de la nécessité de disposer de ressources matérielles pour faire fonctionner correctement le(s) *cluster(s)*. De plus, ces technologies ouvertes évoluent très rapidement, et il est nécessaire de faire une veille technologique pour suivre les nouvelles évolutions, mais aussi de maintenir le *cluster* à jour, ce qui prend du temps et peut parfois être une tâche compliquée. Enfin, dans notre vision du lac de données, nous considérons qu'un écosystème Apache est le bienvenu lorsqu'il est nécessaire de gérer de grandes quantités de données, mais cela n'a pas été un problème dans notre cas d'usage.

L'intégration d'une zone de stockage HDFS dans le lac de données, ainsi que de certains services associés, pourrait cependant être une problématique intéressante à explorer. Pour rappel, le lac de données ArchaeoDAL introduit dans le chapitre 3 est basé sur un écosystème Apache [Liu et al., 2021].

4.5.1.4 Assemblage de technologies ouvertes

Enfin, la dernière piste étudiée est de développer le lac de données *from scratch*, c'est-à-dire de prendre des outils et des technologies différents et de les assembler pour former un lac de données. Ces technologies sont généralement des systèmes de gestion de bases de données NoSQL, qui sont amenés à communiquer entre eux avec des services web développés expressément pour.

Bien que ce soit un pari risqué, c'est la solution que nous avons choisie, notamment pour avoir le plus de flexibilité possible dans la mise en place du système de métadonnées, mais aussi pour rester le plus fidèle possible à notre vision du lac de données. Utiliser des

technologies déjà existantes et/ou faire partie d'un écosystème signifiait que nous courions le risque de voir ces outils évoluer dans le temps, éventuellement dans une direction qui pourrait un jour ne plus correspondre à notre vision du lac de données. Par ailleurs, développer un système de A à Z nous donne la possibilité d'en avoir un contrôle plus complet et de pouvoir le faire évoluer comme nous le souhaitons, en fonction des futurs cas d'usage auxquels nous pourrions être confrontés. Enfin, compte tenu du volume limité de données dans notre cas d'usage, l'utilisation de technologies relativement simples était le meilleur choix à faire.

4.5.2 Présentation des composants de HOUDAL

HOUDAL (*public HOUsing DAta Lake*), notre implémentation d'un lac de données pour l'habitat social, est basée sur une application web. Elle se compose de deux parties principales, avec d'une part le *front-end* (ou partie client) avec laquelle l'utilisateur interagit pour créer, consulter et interroger les métadonnées du lac. D'autre part, le *back-end* (ou partie serveur) se décompose en différents services, à savoir une API (*Application Programming Interface*), le système de métadonnées et le stockage des données.

La figure 4.9 présente les différents composants de HOUDAL et leurs interactions. Nous listons aussi, pour chaque service, les technologies utilisées pour l'implémentation. L'utilisateur interagit avec l'interface, qui elle-même communique avec l'API. C'est ensuite cette dernière qui interroge les différents services en fonction des demandes formulées par l'utilisateur. Nous détaillons ce mode de fonctionnement ci-après.

Par rapport à l'architecture de lac de données proposée dans ce chapitre (figure 4.1), l'interface utilisateur implémente les composants d'ingestion et d'exploitation, puisque c'est avec celle-ci que l'utilisateur va insérer les données dans le lac, puis les récupérer pour un besoin donné. L'API et les autres services implémentent les deux autres composants, qui sont la zone de stockage des données ainsi que le système de gestion des métadonnées du lac.

4.5.2.1 Interface utilisateur

Le lac de données HOUDAL est exploité via une interface utilisateur. À travers celle-ci, l'utilisateur peut consulter et explorer les métadonnées existantes, avec notamment une barre de recherche. Les fichiers stockés dans le lac peuvent être téléchargés. Par ailleurs, il est également possible pour l'utilisateur de créer de nouvelles métadonnées, mais aussi d'ajouter de nouveaux fichiers au lac de données. Pour cela, un formulaire est disponible, avec lequel l'utilisateur téléverse le fichier. Au cours de ce processus, les métadonnées sont saisies de manière semi-automatique : certaines informations, comme le nom, la taille ou le format du fichier, sont extraites automatiquement, tandis que d'autres doivent être saisies manuellement par l'utilisateur, comme une description. Une fois le téléversement du fichier validé, les données sont enregistrées dans la zone de stockage, et les métadonnées sont créées. D'un point de vue technique, l'interface utilisateur a été développée avec la librairie JavaScript nommée ReactJS [3].

3. `https://reactjs.org/`

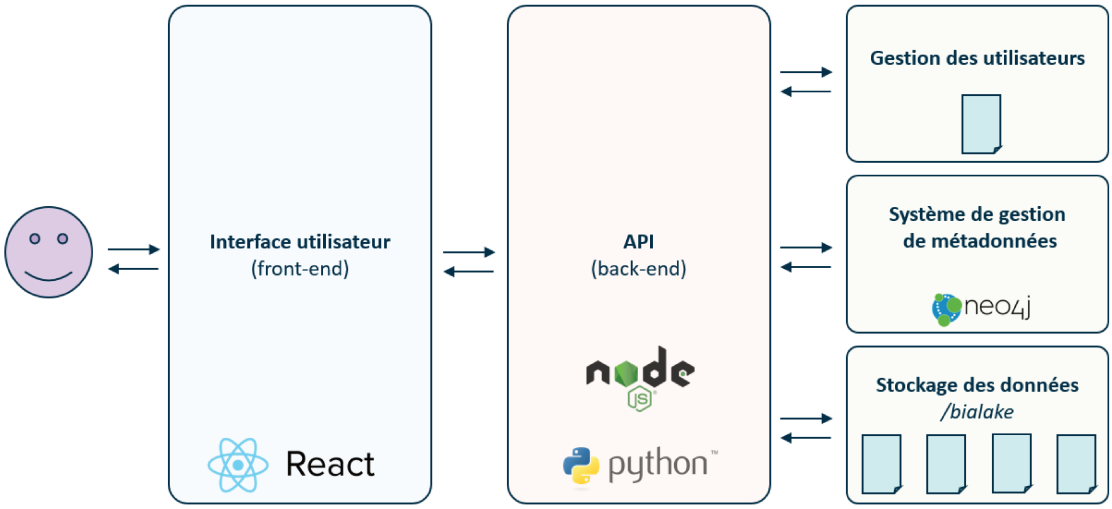

FIGURE 4.9 – Agencement des services de HOUDAL

Une capture d'écran de l'interface utilisateur de HOUDAL est présentée dans la figure 4.10. Les différents boutons situés en haut de l'écran redirigent l'utilisateur vers des pages contenant des formulaires permettant de créer des métadonnées. En outre, l'interface fournit une barre de recherche, où l'utilisateur peut effectuer une recherche par mot-clé dans les entités de données. Sur la capture d'écran, l'utilisateur possède le nom `esy` et a recherché le mot-clé « sollicitation ». L'interface renvoie alors à l'utilisateur toutes les entités de données contenant ce mot-clé dans leur nom de fichier ou leur description.

Notons que la capture d'écran de la figure 4.10 n'est que partielle et ne montre pas toutes les fonctionnalités de l'interface, comme le téléchargement de données ou la visualisation des groupes liés à une entité de données.

4.5.2.2 API

En effectuant des actions sur l'interface, l'utilisateur génère des requêtes qui sont envoyées à l'API. Cette dernière sert « d'aiguillage » entre l'interface utilisateur et les différents services (système de gestion de métadonnées, couche de stockage...). Lorsque l'API reçoit la requête, elle interroge alors les services concernés, puis récupère leurs réponses. Ensuite, elle renvoie la réponse à la requête initiale à l'interface. Cette API a été développée avec NodeJS [4], une plateforme logicielle basée sur JavaScript. Notons que certains traitements de données plutôt volumineux, détaillés dans le chapitre 5, sont effectués en Python.

De plus, HOUDAL fonctionne avec un système d'identification : l'utilisateur doit se connecter pour accéder au lac de données. Par ailleurs, certaines pages de l'interface sont

4. `https://nodejs.org/`

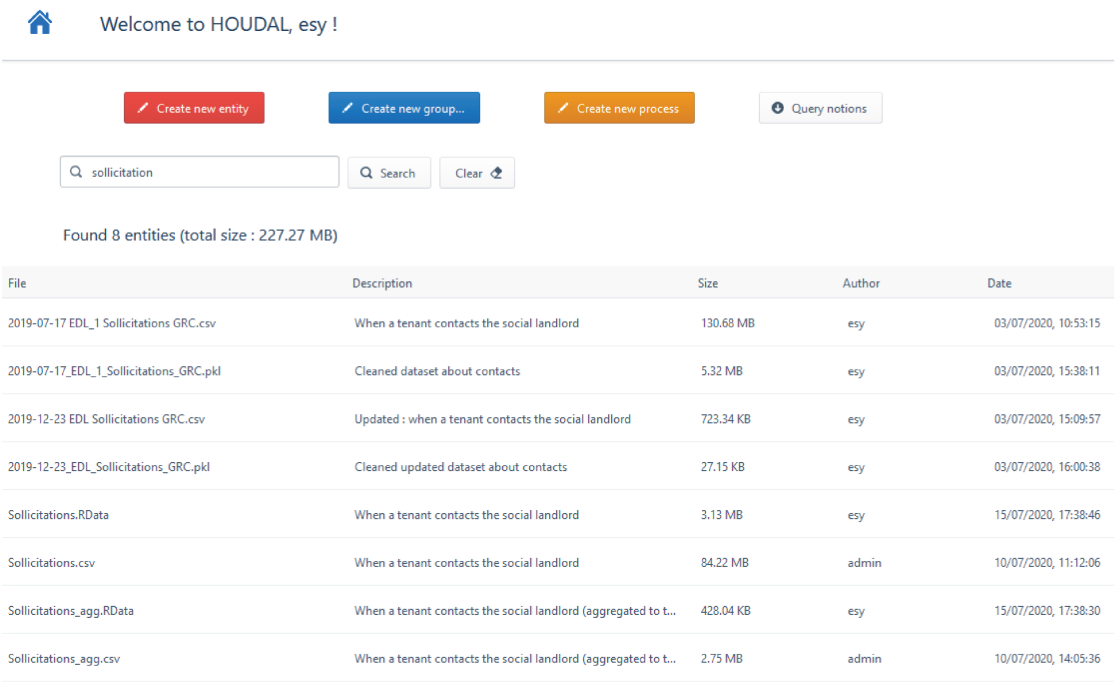

FIGURE 4.10 – Interface utilisateur de HOUDAL

réservées aux utilisateurs ayant un rôle d'administrateur. Les informations concernant les utilisateurs sont enregistrées dans un fichier.

4.5.2.3 Système de gestion de métadonnées

Nous avons présenté en détail dans la section 4.4 le système de gestion de métadonnées (SGMD) de HOUDAL, qui se base sur notre métamodèle goldMEDAL. Puisque le modèle logique instancié est basé sur la théorie des graphes, nous utilisons le SGBD graphe Neo4j pour l'implémentation du SGMD. Cette base de données est accessible uniquement par l'API, que ce soit pour consulter les métadonnées existantes ou pour en créer de nouvelles. Avoir toutes les métadonnées dans une seule base de données est un avantage car cela simplifie grandement les appels d'API. D'un point de vue comptable, il existe 154 nœuds dans Neo4j pour représenter l'entièreté des métadonnées du lac.

4.5.2.4 Stockage des données

Dans notre cas d'usage, nous n'avons que des fichiers à enregistrer, qui sont de taille modérée. Les 110 entités de données du projet EDL représentent un total de 4 gigaoctets de données. Ceci implique que pour implémenter la couche de stockage de HOUDAL, nous n'avions pas besoin d'autre chose qu'un répertoire d'un système de fichiers classique pour stocker nos données.

Notons toutefois que le fait de n'avoir qu'un système de fichiers classique est certes suffisant pour répondre aux besoins du cas d'usage, mais présente néanmoins des limites qui pourraient être atteintes dans d'autres cas d'usages. Ce point est abordé plus en détail dans la sous-section suivante.

4.5.3 Travaux d'amélioration de HOUDAL en cours

HOUDAL est fonctionnel et est actuellement en phase de test avancé. Plusieurs bailleurs sociaux ont montré de l'intérêt pour notre travail et, bien que beaucoup reste à faire, des discussions sont en cours pour améliorer l'application et pour la déployer chez des bailleurs sur le long terme.

4.5.3.1 Stockage des données

La première limite de HOUDAL concerne son système de stockage des données. Comme nous l'avons présenté plus haut, compte tenu du cas d'usage que nous avons traité, avoir un système de stockage autre qu'un répertoire n'était pas nécessaire. En revanche, à terme, nous souhaiterions ajouter à HOUDAL d'autres modes de stockage des données, dans l'optique d'optimiser les performances du lac. En effet, des données tabulaires comme des fichiers CSV pourraient être enregistrés dans une base de données relationnelle, tout comme des documents semi-structurés tels que des fichiers JSON seraient susceptibles d'être conservés dans un SGBD NoSQL orienté documents.

Pour remédier à ce problème, nous souhaitons ajouter des SGBD relationnels et/ou NoSQL dans l'espace de stockage du lac de données. Afin de profiter de leur potentiel, nous voulons également proposer au travers de l'interface une page permettant à l'utilisateur de saisir des requêtes (SQL ou autre), afin d'interroger les différentes bases de données.

4.5.3.2 Ingestion des données

La deuxième limite se situe au niveau du remplissage du lac de données avec les données sources. Bien que ce processus soit assez simple, il reste fastidieux dans le cas d'un grand nombre de fichiers à ingérer. De plus, en fonction de la nature des données à ingérer, il est possible que le processus se complexifie. Si nous prenons l'exemple d'une source qui est une base de données relationnelle, mais que l'utilisateur a besoin d'un niveau de granularité où une entité de données correspond à une table, il doit alors exporter les tables comme des fichiers CSV, puis les insérer un à un dans le lac de données. Avec cet exemple, nous constatons que le processus d'ingestion des données peut devenir très long et redondant pour l'utilisateur.

C'est pourquoi nous souhaitons ouvrir le champ des possibles concernant les sources de données pouvant alimenter le lac de données. En particulier, nous voulons pouvoir automatiser le remplissage du lac. Pour cela, nous voulons développer des connecteurs qui permettent au lac de faciliter ce travail à l'utilisateur. En reprenant l'exemple donné ci-dessus, le lac pourrait se connecter à la base de données et automatiquement récupérer

toutes les tables présentes dans la base, l'utilisateur n'ayant alors qu'à saisir les métadonnées. L'ingestion des données serait alors bien plus rapide que de devoir exporter chaque table puis l'insérer manuellement dans le lac.

En ce qui concerne l'ingestion des données au sein du lac, nous avons mené des travaux pour assister l'utilisateur lors de la création de métadonnées spécifiquement dédiées aux données structurées. Ces travaux sont présentés dans le chapitre 5.

4.5.3.3 Sécuriser l'accès aux données

Le lac de données peut contenir des données de tous types et provenant de toutes origines. De plus, des utilisateurs aux profils différents (spécialistes des données, experts métier ou décideurs, par exemple) sont amenés à se connecter au lac pour l'exploiter. Cependant, certaines données peuvent être de nature sensible et ne doivent être accessibles que par une catégorie spécifique d'utilisateurs. Cet aspect rejoint les problématiques inhérentes au Règlement Général sur la Protection des Données (RGPD). Par exemple, dans le cas de données concernant le logement social, si un ensemble de données contient des détails sur les revenus des locataires ou leur situation professionnelle, il doit être considéré comme sensible et donc uniquement accessible par des utilisateurs accrédités.

Dans le projet EDL, il n'y avait pas de notion de données sensibles dans le lac de données, et seuls des spécialistes des données avaient besoin d'accéder aux données du lac. Nous souhaitons néanmoins travailler sur cet aspect pour de potentiels nouveaux cas d'usage qui présenteraient ce cas de figure. Pour cela, nous voulons introduire les notions de rôles et de restrictions dans l'application de gestion et d'utilisation du lac de données. Nous souhaitons que les rôles soient modulaires, c'est-à-dire qu'il soit possible de définir les rôles pour un projet donné, et d'associer des restrictions à ces rôles. Il est également possible de restreindre l'accès aux métadonnées selon un profil : pouvoir seulement consulter, pouvoir créer de nouvelles métadonnées, pouvoir les modifier, et ainsi de suite.

4.5.4 Réponses aux problèmes du projet EDL

Dans la section 4.2.2, nous avions mentionné quatre problèmes rencontrés par les collaborateurs de BIAL-X lorsqu'ils avaient mené le projet EDL avec des outils traditionnels, i.e. sans lac de données. Nous reprenons ces problèmes et étudions comment HOUDAL apporte des solutions pour les résoudre, ou a minima facilite le travail de l'utilisateur.

Le projet EDL a généré des données aux formats multiples (données structurées, images, tableaux de bord, documents de suivi, etc.). Emmagasiner des données hétérogènes dans leur format natif est justement l'une des principales forces d'un lac de données, et donc de HOUDAL. L'utilisateur n'a donc pas à se soucier de la variété des données qu'il génère, puisque tout peut être enregistré dans le lac.

De plus, plusieurs types d'analyse ont été effectuées sur les données du projet EDL, allant du *reporting* BI aux modèles prédictifs. À nouveau, le lac de données HOUDAL permet de mener tous types d'analyse sur les données qu'il contient. L'utilisateur peut

donc préparer les données, puis alimenter un tableau de bord ou encore un modèle prédictif, selon ses besoins.

Dans le projet EDL, le lignage des données n'a pas été correctement suivi. Grâce au système de métadonnées, et en particulier au concept de processus issu de goldMEDAL, l'utilisateur peut conserver toutes les informations relatives au lignage des données. Si les transformations effectuées sur les données sont des scripts, il est même possible d'enregistrer le fichier du script dans le lac, pour conserver à la fois les informations sur les transformations effectuées, mais aussi le script en lui-même.

Finalement, le dernier problème relevé lors du projet EDL était la difficulté de faire un transfert de connaissance, i.e. d'expliquer simplement le déroulé du projet, et savoir quelles données sont exploitées par quoi, comment et pourquoi. Le système de métadonnées de HOUDAL répond (au moins partiellement) à ce problème. Les entités de données peuvent être liées entre elles, ou rassemblées dans des groupes qui peuvent désigner des catégorisations métier, facilitant la compréhension des données et de leur contenu. De même, comme évoqué, les processus permettent de retracer le lignage des données, permettant de suivre l'évolution des données à travers les traitements. Ainsi, l'utilisateur peut consulter toutes ces métadonnées, et cela constitue un bon socle permettant d'effectuer un transfert de connaissances efficace.

4.6 Conclusion

L'émergence des mégadonnées bouleverse les usages de tous les acteurs de l'activité économique, ce qui inclue les bailleurs sociaux. Un cas d'usage métier rencontré chez BIAL-X illustre parfaitement ces nouveaux challenges. En effet, devoir entreposer des données de tous types, mener diverses analyses sur ces données variées, mais aussi suivre le lignage des données au fil des travaux et pouvoir transférer aisément tous les travaux à des personnes externes au projet sont des problèmes qui ont été rencontrés dans le projet EDL. Pour pallier ces difficultés, nous proposons d'utiliser un lac de données, qui peut stocker des données de tous types, les rendre disponibles pour tous types d'analyses, mais aussi décrire les données ainsi que leurs évolutions à travers des métadonnées. Un lac permet donc, en théorie, de gérer toutes les problématiques identifiées dans notre cas d'usage métier.

C'est pourquoi nous nous sommes penchés sur la mise en œuvre concrète d'un lac de données, en commençant par définir son architecture. Une étude des différentes architectures de lac de données proposées dans la littérature nous pousse à remarquer qu'elles ont toutes tendance à compartimenter les données du lac. Face à ces limites, nous proposons une architecture qui est basée sur quatre composants principaux. En plus de la couche de stockage des données et du système de gestion de métadonnées, nous ajoutons en amont le composant d'ingestion, par lequel passent toutes les données à insérer dans le lac, et qui permet à l'utilisateur de créer les métadonnées associées aux données à insérer. En aval, nous trouvons la couche d'exploitation des données, avec laquelle l'utilisateur peut retravailler et analyser les données présentes dans le lac.

Les lacs de données conventionnels fonctionnent en ELT (pour *Extract - Load - Trans-*

form) ; nous préférons ajouter deux étapes pour obtenir un mode de fonctionnement EDLT(L). L'étape D (pour *Describe*), gérée dans le composant d'ingestion, insiste sur la nécessité de systématiquement créer les métadonnées avant de stocker les données dans le lac. Ceci nous permet de définir une nouvelle propriété pour caractériser les lacs de données : *describe-on-write*. L'étape facultative L, pour *(Re)Load*, décrit les analyses effectuées sur les données du lac qui peuvent générer de nouvelles données retravaillées. Lorsque c'est le cas, ces données sont alors rétroalimentées dans le lac, et passent à nouveau par l'étape *Describe*. Avec ce fonctionnement, nous considérons que le lac de données est un système « vivant » qui peut s'auto-alimenter.

Le métamodèle de métadonnées goldMEDAL est utilisé pour mettre en place le système de gestion de métadonnées de HOUDAL. Nous instancions le métamodèle conceptuel en un modèle conceptuel adapté au cas d'usage, puis nous traduisons ces concepts au niveau logique à l'aide de la théorie des graphes. Finalement, nous traduisons le modèle logique instancié en un modèle physique avec la base de données graphe Neo4j.

Pour implémenter l'architecture de HOUDAL, nous avons choisi de développer une application web « *from scratch* », afin de répondre au mieux aux exigences du cas d'usage. Nous avons donc assemblé des technologies ouvertes, avec en particulier le SGBD graphe Neo4j qui nous sert à enregistrer les métadonnées. HOUDAL est aussi composé d'une interface utilisateur et d'une API qui interroge les différents services nécessaires au bon fonctionnement du lac, en particulier la zone de stockage du lac de données et le système de métadonnées.

HOUDAL montre des résultats satisfaisants, mais nous avons plusieurs points à améliorer pour le rendre plus complet. Nous souhaitons notamment proposer plusieurs méthodes de stockage des données. De plus, nous avons pour objectif d'améliorer l'alimentation du lac de données pour le rendre capable de se connecter à différentes sources, dans l'optique de faciliter le travail de l'utilisateur. Enfin, nous souhaitons également rendre HOUDAL plus robuste face aux problématiques relatives au RGPD, que nous n'avons pas eu à traiter dans notre cas d'usage, mais qui sont néanmoins très courantes.

En marge de ces axes d'améliorations, une nouvelle problématique métier portant sur les données structurées générées de manière périodique a inspiré de nouveaux travaux sur HOUDAL, qui seront présentés en détail dans le chapitre 5.

Chapitre 5

QSTR, assistant à la création de métadonnées pour données structurées

Sommaire

"The principle, V. It's always about the principle. Swap meat for chrome, live a BD fantasy, whatever - but at the end of it all, it's the code you live by that defines who you are. Ever get lost, it shows you the way home. Bust up into pieces, it puts you back together again."

Johnny Silverhand, *Cyberpunk 2077*

Résumé

Les bailleurs sociaux sont amenés à générer de manière périodique certains fichiers de données structurées pour lesquels le schéma est défini par le gouvernement. Toutefois, d'une occurrence à l'autre, le schéma des données est susceptible d'évoluer légèrement. Ces différentes occurrences de fichiers de données sont enregistrées dans le lac de données, donc cela ne pose pas de problème à l'insertion puisque les données sont enregistrées dans le lac sans schéma prédéfini. En revanche, les évolutions de schéma peuvent s'avérer problématique si l'utilisateur doit interroger toutes les occurrences des données en même temps. C'est pourquoi nous cherchons à utiliser le système de gestion des métadonnées du lac pour qualifier au mieux les données à leur insertion, de manière à pouvoir suivre les évolutions subtiles de schéma au fil des occurrences de données, et ainsi faciliter le travail d'interrogation des données pour l'utilisateur. Toutefois, la création de ces métadonnées décrivant l'évolution des données au fil des occurrences est loin d'être une tâche aisée pour l'utilisateur.

Dans ce chapitre, nous proposons QSTR (Qualifying STRuctured data), un assistant à la création de métadonnées dédiées aux données structurées générées périodiquement. QSTR effectue des suggestions de liens entre les attributs des différentes occurrences de données, que l'utilisateur doit ensuite valider ou invalider. Les suggestions prennent appui sur trois critères, qui sont déterminés à l'aide de méthodes statistiques permettant d'extraire des informations des différents attributs, ce qui rend possible les comparaisons entre eux. Finalement, nous modélisons ces nouvelles métadonnées à l'aide du métamodèle goldMEDAL, et nous montrons que nous arrivons à gérer le fait d'avoir des entités de données à des niveaux de granularité différents.

5.1 Introduction

Grâce à nos précédents travaux, nous disposons d'une implémentation d'un lac de données pour l'habitat social, baptisé HOUDAL, que nous avons présenté en détail dans le chapitre 4. En outre, le système de gestion des métadonnées de HOUDAL est basé sur le métamodèle de métadonnées générique goldMEDAL, dont une présentation exhaustive a été faite dans le chapitre 3. Puisque c'est l'une des particularités du cas d'usage sur l'habitat social, le lac est majoritairement rempli de données structurées. Ainsi, notre attention s'est également portée sur l'amélioration de l'expérience utilisateur lors de l'exploitation de ces données structurées au sein du lac de données, que ce soit en amont (insertion) ou en aval (interrogation).

Plus spécifiquement, nous nous sommes heurtés à une problématique métier lors de l'utilisation du lac. Pour certains ensembles de données structurées, les bailleurs sociaux sont tenus de respecter un standard gouvernemental qui impose le schéma des données. De plus, les fichiers sont générés de manière récurrente, en général mensuellement ou annuellement selon les demandes du gouvernement, mais le standard est susceptible d'évoluer à chaque nouvelle échéance. Ces évolutions peuvent parfois entraîner des modifications dans le schéma auquel les données doivent se conformer : un attribut peut changer d'intitulé, un nouvel attribut peut apparaître, ou bien le codage des données d'un attribut peut évoluer.

Dans un lac de données, les données sont enregistrées sans schéma prédéfini, c'est-à-dire que les variations de schéma au fil des occurrences de données ne posent pas de problème à l'ingestion [Sawadogo et al., 2019b]. Toutefois, l'utilisateur peut quand même être amené à devoir interroger toutes les occurrences des fichiers de données de manière simultanée, et donc les différences de schéma peuvent poser problème à ce moment-là. Mais modifier les données avant leur enregistrement dans le lac de données serait contraire à son mode de fonctionnement [Miloslavskaya and Tolstoy, 2016].

Pour pallier ce problème, nous pouvons tirer parti du processus d'ingestion des données, au cours duquel des métadonnées sont créées pour décrire les données. L'utilisateur peut en effet créer, en plus des métadonnées universelles saisies pour tous types de données, de nouvelles métadonnées dédiées spécifiquement à ces données structurées générées de manière périodique, mais dont le schéma est susceptible d'évoluer au fur et à mesure des occurrences de données [Farrugia et al., 2016]. L'objectif de ces métadonnées est de décrire ces évolutions de schéma, ce qui permettra alors à l'utilisateur d'être informé lors de l'interrogation des données que des différences de schéma existent. Avec cette connaissance, si l'utilisateur souhaite interroger plusieurs occurrences de données simultanément, les métadonnées l'aideront alors considérablement pour obtenir un schéma unique. Ces métadonnées peuvent également être utiles à d'autres utilisateurs, qui n'ont ni inséré les données dans le lac ni créé les métadonnées, mais qui pourraient tout de même en avoir besoin pour un besoin donné.

En revanche, le processus de création de ces métadonnées spécifiques pour données structurées peut s'avérer être un travail très fastidieux pour l'utilisateur [Zhao et al., 2021]. En effet, il n'est pas rare que les standards gouvernementaux imposés aux bailleurs

manquent de clarté, et/ou requièrent la présence de nombreux attributs. Décrire d'une manière claire et compréhensible les évolutions de schéma d'une occurrence des données à l'autre n'est pas une tâche aisée. C'est pourquoi nous souhaitons assister l'utilisateur dans cette démarche de création de métadonnées pour données structurées « répétitives ».

Dans ce chapitre, nous proposons QSTR (*Qualifying STRuctured data*), un assistant pour la création des métadonnées spécifiques aux données structurées générées périodiquement. Son objectif est de faciliter le travail de l'utilisateur lors de l'ingestion des données structurées dans le lac. QSTR suggère à l'utilisateur des potentiels liens entre les attributs d'une nouvelle occurrence des données aux attributs des fichiers déjà présents dans le lac, et ce à l'aide de méthodes statistiques. Pour chaque attribut du fichier à insérer dans le lac, l'assistant QSTR calcule des critères statistiques qui permettent de le comparer à un ou plusieurs attribut(s) d'une précédente occurrence des données déjà présente dans le lac. Si les informations suggérées par QSTR sont validées par l'utilisateur, alors elles sont modélisées par les concepts du métamodèle goldMEDAL et implémentées dans le lac. Les concepts de goldMEDAL sont suffisamment génériques pour permettre d'intégrer de telles métadonnées et pour permettre d'avoir des entités de données à des niveaux de granularité différents, c'est-à-dire à la fois au niveau des attributs, mais aussi au niveau des fichiers de données.

Ce chapitre est organisé comme suit. Tout d'abord, nous présentons dans la section 5.2 la problématique métier en détail, puis nous introduisons du vocabulaire pour qualifier les évolutions de schéma des données structurées. Nous détaillons ensuite, dans la section 5.3, la démarche pour extraire des informations servant à comparer des attributs issus de différentes occurrences de données. Après cela, nous présentons dans la section 5.4 QSTR, l'assistant à l'utilisateur pour la création de ces métadonnées spécifiques aux données structurées. Ensuite, la section 5.5 présente comment ces informations sont modélisées avec le métamodèle de métadonnées goldMEDAL, et comment nous arrivons à gérer plusieurs niveaux de granularité au sein de celui-ci. Finalement, la section 5.6 conclut ce chapitre.

5.2 Présentation du problème métier

Dans certains cas de figure, il est possible que l'utilisateur ajoute des données structurées dans le lac de données, qui sont créées de manière périodique et dont il est possible de lier les différentes occurrences entre elles. Nous aimerions alors créer des métadonnées supplémentaires, venant en complément des métadonnées « usuelles », servant à décrire spécifiquement de potentielles évolutions subtiles de schéma entre les différentes occurrences des données structurées générées périodiquement.

Pour traiter ce problème métier, nous avons besoin d'introduire du vocabulaire pour gérer les différents aspects qui en découlent. Tout ceci nous permettra ensuite d'orienter l'extraction d'informations pour la création de métadonnées. Nous commençons par détailler le problème métier que nous souhaitons adresser, ainsi que ses enjeux associés, dans la section 5.2.1. La section 5.2.2 introduit du vocabulaire pour qualifier le problème et présenter des exemples. Finalement, la section 5.2.3 discute des limites de notre approche

en donnant des cas de figure non pris en compte.

5.2.1 Description du problème et enjeux associés

Le problème que nous souhaitons adresser concerne une certaine utilisation du lac de données, lors de la création de métadonnées à l'insertion de données structurées, qui sont générées de manière périodique, et dont il est possible d'établir des liens entre les différentes occurrences des données. Ce problème a été rencontré en travaillant avec les bailleurs sociaux, pour qui certains fichiers de données doivent se conformer à un standard gouvernemental défini chaque année, mais dont le schéma demandé évolue de manière légère à chaque nouvelle occurrence. Par exemple, les bailleurs doivent remplir chaque année un fichier d'enquête contenant des informations sur leur patrimoine ; mais l'enquête diffère légèrement chaque année.

Plus concrètement, d'une année à l'autre, il peut arriver que l'intitulé d'un attribut soit modifié, ou que le codage des données d'un attribut évolue. Toutefois, ces modifications de schéma n'altèrent pas pour autant l'information portée par les attributs, d'un point de vue sémantique. Les bailleurs souhaiteraient pouvoir prendre de la hauteur par rapport à ces différences de schéma pour interroger plusieurs occurrences des mêmes données de manière simultanée, sans avoir à gérer manuellement les différences de schéma. Par exemple, un bailleur pourrait vouloir obtenir « la superficie des logements » pour toutes les occurrences des données, même si des différences peuvent exister entre les attributs qui portent l'information sur la superficie des logements (intitulés différents, par exemple). Pour illustrer, dans une occurrence de données, l'attribut portant cette information est intitulé `surface_habitable`, tandis que dans une seconde occurrence des données, l'attribut se nomme `surf_hab`. Ces deux attributs sont néanmoins porteurs de la même information, à savoir la superficie des logements.

C'est là que des métadonnées peuvent intervenir pour répondre à ce besoin. Puisque les données sont insérées dans le lac de données dans leur état brut, aucune modification n'est effectuée (par exemple, renommer l'intitulé d'un attribut) pour que les données se conforment à un schéma prédéfini. L'utilisateur peut en revanche créer des métadonnées pour décrire les différences de schéma d'une occurrence à l'autre, ce qui facilite ensuite grandement l'interrogation de plusieurs occurrences des données, puisque les différences entre les attributs sont décrites.

Ce travail de description des évolutions du schéma des données au fil des occurrences est ici effectué à l'ingestion des données dans le lac de données. En revanche, la modification concrète des données pour que toutes les occurrences soient conformes au même schéma n'est faite qu'à l'interrogation, si un besoin l'exige. La description et la modification des différences sont donc découplées, contrairement au mode de fonctionnement d'un entrepôt de données, où il faut à la fois comprendre les différences entre les occurrences des données, puis effectuer les modifications immédiatement avant de sauvegarder les données dans l'entrepôt.

Ainsi, peu importe le système employé, l'utilisateur doit systématiquement effectuer ce travail de description des différences de schéma entre les occurrences des données.

L'avantage majeur de disposer d'un lac de données est donc que les données ne sont pas altérées avant leur insertion, donc il n'y pas de perte d'information. De plus, le temps passé pour décrire les différences de schéma n'est pas perdu, car tout est enregistré dans les métadonnées et peut être utilisé ultérieurement.

Toutefois, ce travail de description, et donc de création de métadonnées, peut être un travail fastidieux. En effet, le standard gouvernemental peut par exemple, d'une année à l'autre, imposer un renommage de tous les intitulés d'attributs. Effectuer le lien entre les attributs identiques (mais ne portant pas le même nom) des deux occurrences peut s'avérer être une tâche chronophage et pénible, surtout si les données présentent beaucoup d'attributs. Notre objectif est donc de proposer une aide à l'utilisateur pour faciliter son travail de création de liens entre les différentes occurrences des données, en connectant les attributs entre eux.

5.2.2 Proposition de vocabulaire

Pour traiter cette problématique de création de métadonnées à l'ingestion de données structurées générées périodiquement et dont le schéma est susceptible d'évoluer au fil des occurrences, nous allons utiliser trois termes : attribut, notion et référence. Précisons que, dans la suite du document, nous désignons une occurrence de données comme un exemplaire des fichiers générés périodiquement. Par exemple, pour un fichier d'enquête produit annuellement et ce depuis l'année 2018, l'occurrence 1 correspond au premier fichier (créé en 2018), l'occurrence 2 correspond au second fichier (créé en 2019), etc. Plus globalement, nous pouvons dire que l'occurrence i correspond au fichier i.

Définition 13 *Un* attribut *est une caractéristique présente dans un ensemble de données structurées.*

Notons qu'il est aussi possible d'utiliser les termes champ, colonne ou propriété, mais nous nous référons au terme d'attribut dans la suite du document.

Pour illustrer, le tableau 5.1 ci-dessous présente deux attributs : le premier est intitulé `code_postal`, et le second est nommé `ville`. Nous représentons aussi quelques observations pour chaque attribut.

`code_postal`	`ville`
69760	Limonest
69500	Bron
69009	Lyon 9
01000	Bourg-en-Bresse

TABLE 5.1 – Exemple d'attributs avec quelques observations

Définition 14 *Une* notion *désigne les attributs des différentes occurrences des données qui portent la même information d'un point de vue sémantique, même si l'intitulé et/ou le codage des attributs diffèrent. Une notion est notée* $\nu_i = \{\alpha_{ij}\}_{j \in \mathbb{N}^*}$*, où* α_{ij} *est l'attribut* i *provenant de l'occurrence* j*.*

Soit deux attributs désignant des codes postaux et provenant de deux occurrences de données. Dans la première occurrence, l'attribut porte l'intitulé `code_postal` et dans la seconde occurrence, l'attribut possède comme intitulé `CP`. La notion « Code postal du logement », notée ν_1 rassemble les deux attributs notés respectivement α_{11} et α_{12} : $\nu_1 = \{\alpha_{11}, \alpha_{12}\}$.

Définition 15 *Une* référence *rassemble les notions présentes dans une série d'occurrences de données. Une référence est notée* $\rho = \{\nu_i\}_{i \in \mathbb{N}^*}$.

D'une certaine manière, une référence est le schéma « mappé » de toutes les occurrences des données qui ont été générées au fil du temps, et dont le schéma diffère légèrement à chaque fois. Les notions permettent d'effectuer la liaison entre des attributs d'occurrences de données différentes, qui sont porteurs de la même information, mais dont l'intitulé et/ou le codage peut différer. Une référence est donc un ensemble d'informations, et se place à un niveau d'abstraction un peu plus élevé qu'un schéma classique qui définit un ensemble d'attributs. Pour illustrer, soit trois notions : ν_1 est « Identifiant du logement », ν_2 désigne le « Code postal du logement » et ν_3 correspond à la « Ville du logement ». La référence contenant ces trois notions est notée $\rho = \{\nu_1, \nu_2, \nu_3\}$.

Donnons maintenant un exemple complet pour illustrer l'utilisation de ce vocabulaire. Imaginons qu'un standard gouvernemental impose aux bailleurs de générer annuellement un fichier contenant des informations sur les logements de leur patrimoine, avec comme informations l'identifiant du logement, sa typologie, sa surface habitable, le code postal et le nom de la ville dans laquelle il est situé. Le tableau 5.2 donne un extrait de la première occurrence des données.

`cd_logement`	`typologie`	`surf_hab`	`cd_postal`	`lb_ville`
1	T3	66	69760	Limonest
2	T4	82	69500	Bron
3	T2	48	69009	Lyon 9
4	T3	60	01000	Bourg-en-Bresse

TABLE 5.2 – Attributs et quelques observations de la première occurrence de données sur des logements

Avec cette première occurrence de données sur les logements des bailleurs, une référence « Logement » est créée, et sera utilisée ultérieurement pour capter les futures occurrences des données. Puisque c'est la première occurrence, la référence est « vide », c'est-à-dire qu'il n'existe pas encore de notion : l'utilisateur doit alors créer une notion pour chaque attribut, et rattacher ce dernier à la notion correspondante. Le tableau 5.3 présente la référence « Logement » en listant les différentes notions et en associant à chacune l'attribut correspondant.

L'année suivante, le bailleur doit générer une seconde occurrence des données qui correspond aussi au standard du moment imposé par le gouvernement. Toutefois, par

Notion \ Occurrence	Attributs de l'occurrence 1
Identifiant	`cd_logement`
Typologie	`typologie`
Surface habitable	`surf_hab`
Code postal	`cd_postal`
Libellé ville	`lb_ville`

TABLE 5.3 – Notions de la référence « Logement » après ingestion de la première occurrence des données

rapport à l'année précédente, ce standard a légèrement évolué. Le tableau 5.4 montre les attributs de la seconde occurrence des données, ainsi que quelques observations.

`cd_logement`	`cd_postal`	`lb_ville`	`nb_pieces`	`surface`	`chauffage`
1	69760	Limonest	3	66	C
2	69500	Bron	4	82	C
3	69009	Lyon 9	2	48	I
4	01000	Bourg-en-Bresse	3	60	I
5	69760	Limonest	2	42	I

TABLE 5.4 – Attributs et quelques observations de la seconde occurrence de données sur des logements

Nous constatons plusieurs évolutions entre la première et la seconde occurrence des données :

- L'attribut `typologie` de la première occurrence n'a pas d'équivalent direct dans la seconde occurrence, mais un nouvel attribut `nb_pieces` porte la même information, et peut donc être rattaché à la même notion. Notons que le codage change : les données étaient initialement des valeurs alphanumériques (T2, T3, T4) et deviennent ensuite seulement numériques (2, 3, 4).
- L'attribut nommé initialement `surf_hab` change d'intitulé dans la nouvelle occurrence des données pour devenir `surface`.
- Un nouvel attribut `chauffage` fait son apparition dans la seconde occurrence des données.
- Les attributs `nb_pieces` et `surface` changent de position : initialement aux positions 2 et 3, ils passent aux positions 4 et 5 dans la seconde occurrence.

Le tableau 5.5 illustre la référence « Logement » mise à jour avec la seconde occurrence des données. Les différents attributs ont pu être reliés aux notions existantes, sauf le nouvel attribut `chauffage` qui n'a pas de notion correspondante dans la référence. Ainsi, une nouvelle notion « Type de chauffage » est créée dans la référence, mais qui n'a pas d'attribut associé dans la première occurrence des données.

L'exemple que nous avons déroulé ici pour illustrer est relativement simple et facile à comprendre. Il faut toutefois garder à l'esprit qu'en situation réelle, les fichiers de

Notion \ Occurrence	Attributs de l'occurrence 1	Attributs de l'occurrence 2
Identifiant	`cd_logement`	`cd_logement`
Typologie	`typologie`	`nb_pieces`
Surface habitable	`surf_hab`	`surface`
Code postal	`cd_postal`	`cd_postal`
Libellé ville	`lb_ville`	`lb_ville`
Type de chauffage	—	`chauffage`

TABLE 5.5 – Notions de la référence « Logement » après ingestion de la seconde occurrence des données

données contiennent généralement bien plus d'attributs, que leurs intitulés n'ont pas systématiquement des noms très parlants, ou encore que les données peuvent présenter des problèmes de qualité. Tous ces éléments peuvent considérablement ralentir l'utilisateur dans sa démarche de création de ces métadonnées entre les occurrences des données, en particulier pour rassembler les différents attributs au sein de notions. Ceci souligne l'importance d'un assistant à la création de métadonnées pour ces données structurées générées de manière périodique, afin de faciliter le travail de l'utilisateur.

En outre, disposer de ces métadonnées liant les occurrences de données entre elles facilitent aussi beaucoup l'interrogation des données. Il devient en effet possible d'interroger un schéma de référence à travers ses notions, en formulant une requête comme *« sélectionner dans la référence « Logement » l'identifiant, la typologie et la surface habitable des logements pour toutes les occurrences disponibles »*. À partir de là, le système de métadonnées du lac peut indiquer à l'utilisateur quels attributs sont concernés par les notions demandées, et ce pour chaque occurrence des données. Nous pouvons aussi imaginer, dans un second temps, qu'un système plus avancé pourrait aussi spécifier les différences de codage entre les attributs s'ils existent, et suggérer des opérations simples de modification pour uniformiser les données.

5.2.3 Discussion : cas de figure non pris en compte

Grâce au vocabulaire introduit dans la section précédente, à savoir une référence, contenant des notions, chacune permettant de regrouper des attributs, nous pouvons gérer la question de l'évolution du schéma des données au fil du temps, même si l'évolution se fait sur un temps long. Notre approche présente toutefois quelques limites, et nous en avons relevé deux, bien que d'autres pourraient être listées.

Premièrement, imaginons que pour une référence donnée, chaque occurrence des données voit son schéma évoluer de manière conséquente. Pouvons-nous considérer qu'à un certain stade, une nouvelle occurrence des données ingérée dans le lac de données n'a plus rien à voir avec la toute première occurrence, qui avait créé la référence ? En d'autres termes, la question se pose de savoir si une référence peut devenir obsolète après trop d'évolutions du schéma des données lors des différentes occurrences. Dans ce cas de figure, certaines notions qui avaient été créées à l'ingestion des premières occurrences des données pourraient ne plus servir pour les occurrences les plus récentes. Pour pallier ce

problème, nous considérons que l'utilisateur crée une nouvelle référence lorsque cela est nécessaire, c'est-à-dire lorsqu'il estime que la référence est devenue obsolète par rapport aux nouvelles données en cours d'ingestion au sein du lac.

En outre, il peut arriver que l'information portée par un seul attribut dans une occurrence des données (et donc liée à une seule notion) peut se séparer en deux attributs distincts dans une nouvelle occurrence des données. Par exemple, une notion « Nom et prénom du locataire » peut devenir obsolète car dans la nouvelle occurrence des données, il existe deux attributs distincts pour le nom et le prénom, ce qui aurait pour effet de générer deux nouvelles notions. Le phénomène inverse peut aussi arriver, à savoir que l'information portée par deux attributs se retrouve dans un seul attribut dans une nouvelle occurrence des données. En inversant l'exemple précédent, une référence contient deux notions « Nom » et « Prénom », mais lors de l'ingestion d'une nouvelle occurrence des données, un attribut contenant à la fois le nom et le prénom fait son apparition, forçant l'utilisateur à créer une troisième notion dédiée. Ce cas de figure est intéressant mais complexe à prendre en compte. Pour le premier exemple (fission d'un attribut en plusieurs), nous pourrions imaginer que pour une occurrence des données, une notion puisse rassembler plusieurs attributs au lieu d'un seul. En revanche, pour le second exemple (fusion de plusieurs attributs en un), il faudrait qu'un attribut soit lié à plusieurs notions simultanément. Il serait toutefois intéressant de chercher à modéliser cela dans des travaux futurs.

5.3 Extraction d'informations pour la création de métadonnées

Maintenant que nous avons présenté en détail les enjeux et les objectifs de ces travaux, nous pouvons désormais nous pencher sur l'extraction d'informations spécifiques permettant de créer des métadonnées sur les données structurées générées périodiquement. Plus précisément, ces informations portent sur les attributs, et rendent possible des comparaisons entre eux. Nous introduisons trois critères pour comparer deux attributs et indiquer à l'utilisateur s'ils semblent similaires ou non. S'ils sont similaires, ils sont alors susceptibles d'être liés à la même notion au sein d'un schéma de référence, et nous souhaitons dans ce cas suggérer à l'utilisateur de créer un lien entre ces deux attributs en les rassemblant au sein de la même notion.

Nous commençons par présenter dans la section 5.3.1 comment le vocabulaire introduit précédemment intègre les comparaisons entre attributs. Après cela, nous explicitons notre méthode pour distinguer la nature des attributs dans la section 5.3.2. Ensuite, nous présentons les trois critères qui nous servent à comparer les attributs entre eux. Le premier est basé sur l'intitulé des attributs (section 5.3.3), le second sur la proportion de valeurs communes (section 5.3.4) et le troisième sur une valeur statistique (section 5.3.5).

5.3.1 Attribut principal d'une notion

À l'insertion d'une nouvelle occurrence d'un fichier de données structurées au sein du lac, l'utilisateur doit lier les attributs de ce nouveau fichier aux notions déjà présentes dans la référence adéquate. S'il n'y pas de notion correspondant à un attribut, il doit alors en créer une nouvelle.

Cependant, la question se pose de comment comparer un attribut d'une nouvelle occurrence des données à une notion. En effet, une notion peut associer plusieurs attributs entre eux, s'il y a eu plusieurs occurrences de données précédemment. Une première solution serait de comparer le nouvel attribut à tous les attributs qui sont liés à la notion sélectionnée, et ensuite fournir à l'utilisateur par exemple le meilleur résultat, une moyenne, ou l'ensemble des résultats. Toutefois, ce fonctionnement peut vite devenir extrêmement gourmand en ressources dans le cas où beaucoup d'occurrences d'un fichier ont été générées, et donc qu'une notion peut être liée à de nombreux attributs.

Pour cela, nous proposons de désigner pour chaque notion un *attribut principal*, qui est celui qui sera utilisé pour la comparaison avec un nouvel attribut. L'attribut principal sert de représentant pour tous les autres attributs de la même notion. Bien entendu, il est nécessaire de désigner cet attribut principal parmi toutes les attributs de la notion. Nous proposons soit de laisser à l'utilisateur le soin de choisir cet attribut principal, soit de désigner automatiquement l'attribut de la dernière occurrence des données comme étant l'attribut principal, qui représente la notion lors des comparaisons.

5.3.2 Nature des attributs

L'un des critères utilisés pour comparer les attributs entre eux (présenté dans la section 5.3.5) nécessite de connaître la nature des deux attributs comparés. Cependant, puisque la nature des attributs n'est pas connue *a priori*, il est nécessaire de l'inférer à partir du codage des attributs. Nous proposons pour cela une classification dans laquelle nous distinguons cinq possibilités concernant la nature des attributs, en nous basant sur le codage de l'attribut.

La figure 5.1 illustre le procédé permettant de déterminer la nature d'un attribut en fonction du codage de ses données. Sur la partie gauche de la figure, nous voyons le cas où les données d'un attribut comportent des caractères autres que des chiffres. Le premier test effectué est de savoir si le codage des valeurs correspond à des dates, i.e. des valeurs telles que JJ/MM/AAAA, ou équivalent. Si oui, alors la nature de l'attribut est *date*, sinon un second test est effectué. Nous comptons alors le nombre de valeurs distinctes, que nous divisons par le nombre total de valeurs, c'est-à-dire l'effectif total. Si ce ratio est inférieur à 0,1, i.e. s'il y a peu de valeurs distinctes au regard du volume de modalités possibles, alors nous considérons cet attribut comme *catégoriel*, sinon c'est un attribut *textuel*.

La partie droite de la figure décrit le cas où les valeurs de l'attribut ne comportent que des caractères numériques. Ici, un simple test est effectué : si toutes les valeurs sont des 0 et des 1, alors l'attribut est *booléen*. Dans le cas contraire, c'est un attribut *quantitatif*.

Pour dresser un bilan, selon notre classification, un attribut peut ainsi être :

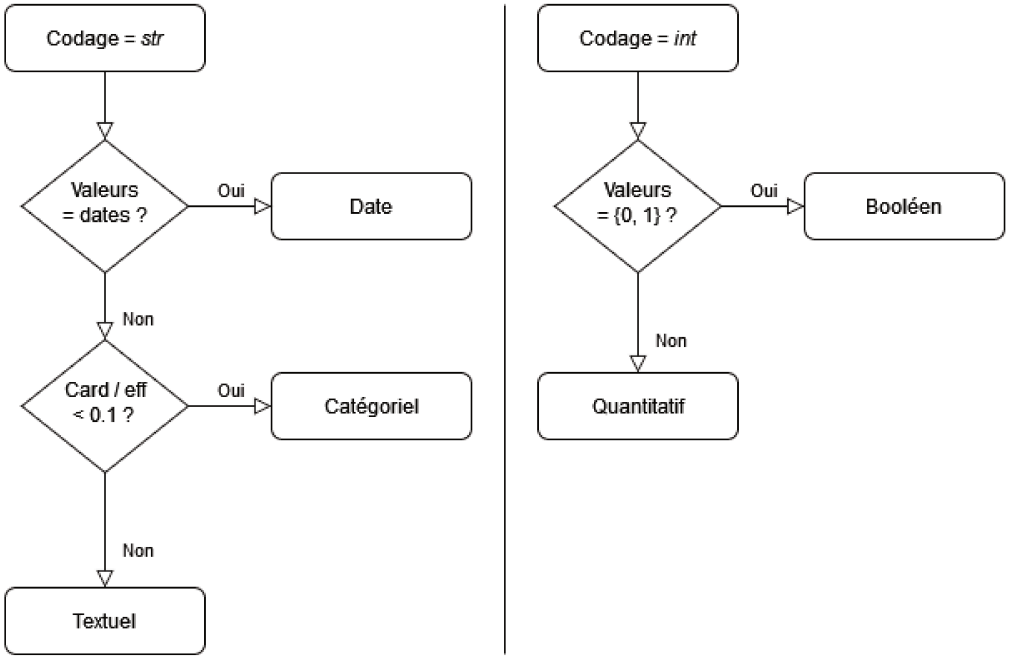

FIGURE 5.1 – Distinction de la nature d'un attribut en fonction du codage de ses valeurs

- **Quantitatif** : les valeurs sont uniquement composées de caractères numériques (exemple : 1, -2, 3,14)
- **Catégoriel** : les valeurs contiennent des caractères, mais le nombre de valeurs différentes est peu élevé (exemple : type de chauffage d'un logement, « individuel » ou « collectif »)
- **Textuel** : les valeurs contiennent des caractères textuels, avec beaucoup de valeurs différentes (exemple : adresse d'un logement)
- **Booléen** : les valeurs sont 0 ou 1
- **Date** : les valeurs représentent des dates (exemple : 01/05/2018)

Inférer la nature d'un attribut à partir de son codage présente toutefois des limites. À titre d'exemple, avec notre démarche, un attribut catégoriel codé en numérique (par exemple, un attribut « sexe » dont les valeurs sont 1 et 2) serait considéré, à tort, comme un attribut quantitatif. Il serait intéressant d'explorer de nouvelles pistes dans des travaux futurs, pourquoi pas à l'aide de méthodes d'apprentissage automatique pour entraîner un modèle à inférer la nature d'un attribut.

5.3.3 Intitulés d'attributs

Le premier critère nous permettant de comparer deux attributs se base sur leur intitulé. Pour les comparer, nous utilisons le ratio de Levenshtein. Cette métrique est basée sur la distance de Levenshtein, appelée aussi la distance d'édition, qui donne une mesure

de la différence entre deux chaînes de caractères [Levenshtein et al., 1966]. La distance de Levenshtein donne le nombre minimal d'opérations requises (à savoir des remplacements, des insertions et des suppressions) sur les caractères d'une chaîne pour la convertir en une autre. Aussi, la formule du ratio de Levenshtein entre deux chaînes de caractères a et b est :

$$ratio_L(a,b) = 1 - \frac{distance_L(a,b)}{|a| + |b|} \tag{5.1}$$

où $|a|$ (resp. $|b|$) est le nombre de caractères de a (resp. b). Le ratio donne un score compris entre 0 et 1 : plus la valeur est proche de 1, plus les deux chaînes de caractères (en l'occurrence, les intitulés d'attributs) sont proches.

Exemple 12 *Soit un attribut α_1 portant comme intitulé* `CD_TYPLOC` *et un attribut α_2 dont l'intitulé est* `TYPOLOGIE`*. La distance de Levenshtein entre les intitulés de α_1 et α_2 vaut 7, et la longueur de chaque intitulé vaut 9. Le ratio de Levenshtein calculé entre α_1 et α_2 est donc : $ratio_L(\alpha_1, \alpha_2) = 1 - \frac{7}{9+9} = 0,61$.*

Ce ratio donne une première indication quant à la proximité de l'intitulé des deux attributs que l'utilisateur souhaite comparer. Il convient toutefois de noter que la proximité d'édition de deux intitulés ne renseignent en rien de la proximité sémantique des attributs. Cela donne seulement une première indication, qui n'est pas suffisante pour conclure quant à la proximité sémantique des deux attributs comparés. Par exemple, ce critère devient caduque lors d'un changement drastique de l'intitulé d'un attribut dans une nouvelle occurrence des données, avec une nouvelle convention de nommage totalement différente. Notons toutefois que dans ce cas de figure, une intervention humaine est forcément nécessaire, généralement à l'aide d'un dictionnaire de données.

5.3.4 Coefficient de superposition

En guise de second critère pour comparer deux attributs, et plus particulièrement leurs valeurs en commun, nous utilisons le coefficient de superposition (*overlap coefficient*), aussi appelé le coefficient de Szymkiewicz–Simpson [Szymkiewicz, 1934]. Pour le calculer, il faut d'abord déterminer l'ensemble des valeurs distinctes de chaque attribut. Ensuite, le coefficient se calcule via le ratio entre la taille de l'intersection des deux ensembles (c'est-à-dire le nombre de valeurs en commun) et la taille du plus petit des deux ensembles (i.e. l'attribut ayant le moins de valeurs distinctes). En notant X (resp. Y) l'ensemble des valeurs distinctes de l'attribut α_1 (resp. α_2), ce coefficient est défini de la manière suivante :

$$overlap(X,Y) = \frac{|X \cap Y|}{min(|X|, |Y|)} \tag{5.2}$$

Exemple 13 *Soit un attribut α_1 ayant comme ensemble de valeurs distinctes $X = \{$ Limonest, Bron, Lyon 9, Bourg-en-Bresse $\}$, et α_2 ayant comme ensemble de valeurs distinctes $Y = \{$ Bron, Limonest, Strasbourg $\}$. La taille de l'intersection de ces deux ensembles est $|X \cap Y| = |\{$ Limonest, Bron $\}| = 2$, et la taille du plus petit ensemble de*

valeurs distinctes est ici $|Y| = 3$. *Alors, le coefficient de superposition calculé entre* α_1 *et* α_2 *est :* $overlap(X, Y) = \frac{2}{3} = 0,66$.

Connaître la proportion de valeurs partagées par deux attributs peut donner une bonne indication quant à leur proximité sémantique, et donc de leur potentiel lien au sein d'une même notion. Néanmoins, cet indice peut aussi être mis en défaut. À titre d'exemple, si les valeurs d'un attribut donnent le nombre de pièces d'un logement (2, 3, 4...), une nouvelle version de cet attribut dans une nouvelle occurrence des données peut cette fois-ci donner la typologie du logement (T2, T3, T4...). Dans ce cas de figure, le coefficient de superposition est nul, bien que les données des attributs soient fortement liées d'un point de vue sémantique. Ainsi, le coefficient de superposition donne une indication mais qui, comme pour le critère précédent, n'est pas suffisante pour conclure sur la proximité sémantique de deux attributs.

Notons que nous aurions aussi pu avoir recours à l'indice de Jaccard, aussi appelé *Intersection over Union (IoU)*. Toujours avec les ensembles de valeurs distinctes X et Y, cet indice donne le ratio entre la taille de l'intersection des deux ensembles (c'est-à-dire, le nombre de valeurs en commun) et la taille de l'union des deux ensembles . Il est ainsi noté :

$$J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{5.3}$$

Toutefois, nous avons préféré écarter cet indice de Jaccard qui a tendance à pénaliser la différence de taille entre les deux ensembles X et Y, et ce même si leur intersection est très grande. En d'autres termes, si les deux attributs présentent énormément de valeurs en commun, mais que l'un des attributs possède beaucoup plus de valeurs distinctes que l'autre, l'indice de Jaccard sera tout de même très faible.

5.3.5 Critère statistique

Le troisième critère que nous utilisons pour comparer deux attributs se base plus en détail sur la distribution des données qu'ils portent. C'est pour ce critère que nous avons besoin de connaître la nature des attributs, que nous déterminons au préalable grâce à la méthode présentée dans la section 5.3.2. En fonction de la nature des deux attributs, il est possible d'utiliser des méthodes comme des ratios ou des corrélations ; dans d'autres cas, nous avons recours à des tests statistiques non-paramétriques, qui ne nécessitent pas d'hypothèse sur la distribution des données.

Nous listons ci-dessous les différents cas de figures identifiés, en fonction de la nature des deux attributs comparés.

- Pour deux attributs **quantitatifs**, nous utilisons le test de Kolmogorov-Smirnov [Massey Jr, 1951]. C'est un test d'hypothèse qui nous permet de déterminer si les observations de deux attributs suivent la même distribution.
- Lorsque deux attributs sont **catégoriels**, ou bien que l'un est **catégoriel** et l'autre **booléen**, ou encore que les deux sont **booléens**, nous utilisons le test d'indépendance du Khi-deux [Pearson, 1900]. C'est un test d'hypothèse, où l'hypothèse nulle

est l'indépendance des attributs. Ainsi, en rejetant l'hypothèse nulle, nous pouvons considérer que les distributions des deux attributs sont liées. Il se base sur les fréquences d'apparition de chaque modalité pour les deux attributs ; dans ce cas de figure, le test est effectué avec 1 degré de liberté.

- Pour deux attributs **textuels**, nous commençons par filtrer les mots vides (*stop-words*) pour chaque attribut, puis nous lemmatisons [Sawadogo et al., 2021]. Ensuite, nous appliquons la méthode *Doc2Vec* [Le and Mikolov, 2014] pour créer un vecteur de dimension réduite pour chaque attribut. Enfin, nous mesurons la similarité Cosinus [Allan et al., 2000] entre ces deux vecteurs pour les comparer.

En marge des cas de figure identifiés ci-dessus, nous pensons que d'autres cas pourraient être intéressants à étudier. Par exemple, si un attribut est quantitatif et l'autre est catégoriel, ou booléen ; ou encore si un attribut est catégoriel et l'autre est textuel. Nous n'avons pas pu nous pencher plus en détail sur ces possibilités par manque de temps, mais il serait intéressant de les traiter dans des travaux futurs. En revanche, les autres cas de figure non mentionnés ne nous paraissent pas intéressants à traiter.

Nous proposons une seconde approche pour le calcul de ce critère. Rappelons que nous travaillons dans cette problématique avec des données dont le schéma est susceptible d'évoluer légèrement d'une occurrence à l'autre, mais qui présentent tout de même plusieurs similitudes. Ainsi, il peut arriver dans certains cas que l'utilisateur soit capable de spécifier une « clé », permettant d'apparier les observations des deux occurrences de données. Cette clé prend la forme d'un attribut qui est présent dans les deux occurrences de données, comme par exemple un identifiant. Pour illustrer, prenons l'exemple d'une référence « Logement » contenant une notion « Identifiant ». Pour cette notion, il existe un attribut dans chaque occurrence des données, et tous les attributs sont identiques. Dans ce cas, il est alors possible d'apparier les données, c'est-à-dire d'indiquer que la clé commune entre les occurrences est l'identifiant du logement.

À nouveau, nous dressons ci-dessous une liste des méthodes statistiques utilisées selon la nature des attributs, cette fois-ci lorsque l'utilisateur peut apparier les données.

- Pour deux attributs **quantitatifs**, nous utilisons le Rho de Spearman [Spearman, 1961]. C'est un coefficient de corrélation entre les rangs des valeurs prises par les deux attributs. Plus la valeur est proche de 1 (ou -1), plus les deux attributs sont considérés comme proches ; à l'inverse, si la valeur s'approche de 0, les deux attributs ne sont pas corrélés.
- Avec deux attributs **catégoriels**, ou bien un attribut **catégoriel** et un autre **booléen**, nous avons recours au test d'indépendance du Khi-deux [Pearson, 1900]. Contrairement à la méthode employée pour des échantillons indépendants, nous ne comparons ici pas les fréquences, mais les valeurs que prennent les deux attributs. À nouveau, c'est un test d'hypothèse, dont l'hypothèse nulle est celle de l'indépendance des deux attributs.
- La distance de Levenshtein nous permet de comparer deux attributs **textuels** [Levenshtein et al., 1966]. De la même manière que pour comparer les intitulés d'attri-

buts, nous cherchons ici à étudier la similarité entre données textuelles au niveau des valeurs des attributs.

- Pour comparer deux attributs **booléens**, nous utilisons l'indice binaire de Jaccard [Jaccard, 1912]. Il y a quatre couples de valeurs possibles : (0,0), (0,1), (1,0) et (1,1). Cet indice calcule le ratio entre le nombre de couples valant (1,1) et le nombre de couples valant (0,1) ou (1,0) ou (1,1). Plus ce ratio est élevé, plus les valeurs des attributs sont considérées comme similaires.
- Dans le cas où sont comparés un attribut **quantitatif** et un attribut **catégoriel**, c'est le test de Kruskal-Wallis qui entre en jeu [Kruskal and Wallis, 1952]. L'hypothèse formulée est que la distribution des attributs est identique. L'idée ici est de rassembler, pour chaque modalité de l'attribut catégoriel, les valeurs de l'attribut quantitatif. Le test d'hypothèse de Kruskal-Wallis est ensuite appliqué en comparant ces ensembles de valeurs quantitatives.
- Enfin, si un attribut est **quantitatif** et un autre est **booléen**, nous avons recours au coefficient de corrélation bisériale ponctuelle [Tate, 1954]. Cette métrique se base sur le coefficient de Pearson, mais dans le cas où l'un des deux attributs est dichotomique, i.e. qu'il ne possède que deux valeurs distinctes (0 ou 1). C'est un coefficient de corrélation, donc si le résultat est proche de 1 ou -1, cela signifie que les attributs sont liés ; si le résultat s'approche de 0, alors les attributs ne sont pas liés.

En plus des cas de figure listés plus haut, nous pensons que le cas où un attribut est catégoriel et le second est textuel pourrait être intéressant à étudier. Comme dans le cas des données non appariées, nous n'avons pas pu explorer cette option par manque de temps. Les autres cas de figure ne nous semblent pas pertinents à traiter.

5.4 QSTR : assistant pour la création de métadonnées

La comparaison des attributs peut s'avérer être un processus fastidieux, surtout lorsqu'une référence contient beaucoup de notions et/ou que les nouvelles données possèdent beaucoup d'attributs, et ce même si l'utilisateur connaît bien ses données. Si ce n'est pas le cas, chercher à faire correspondre un attribut de la nouvelle occurrence des données à la bonne notion d'une référence peut devenir un vrai challenge pour l'utilisateur. C'est dans cette optique que nous proposons QSTR (*Qualifying STRuctured data*), un assistant à la comparaison d'attributs, et donc à la création de métadonnées pour les données structurées générées périodiquement. QSTR vise à faciliter le travail de l'utilisateur, à l'aide des critères présentés dans la section précédente, pour l'aider à déterminer si un nouvel attribut et une notion (représentée par son attribut principal) semblent similaires ou non.

Nous commençons par présenter dans la section 5.4.1 les différents seuils utilisés pour chaque critère, pour renvoyer à l'utilisateur une indication pertinente et utilisable, et ainsi l'aider dans sa prise de décision. Ensuite, nous présentons les deux algorithmes que nous avons développés et qui composent QSTR. Le premier concerne une vérification manuelle

si l'utilisateur a préalablement fait correspondre chaque attribut de la nouvelle occurrence des données aux notions de la référence, et souhaite simplement vérifier les valeurs des critères de comparaison (section 5.4.2). Le second algorithme est une suggestion automatique, où l'utilisateur demande à l'assistant d'essayer de trouver automatiquement les meilleurs couples attribut - notion pour la référence sélectionnée (section 5.4.3). Nous discutons dans la section 5.4.4 des limites actuelles de l'assistant QSTR.

5.4.1 Seuils pour chaque indicateur

Les trois critères présentés dans la section précédente sont basés sur des méthodes différentes. Alors que la comparaison des intitulés des attributs (ratio de Levenshtein) et du taux de leurs modalités communes (coefficient de superposition) fournissent des valeurs comprises entre 0 et 1, le troisième critère est plus complexe à gérer. En effet, la méthode statistique employée diffère en fonction de la nature des attributs comparés, et les résultats obtenus n'ont pas la même forme. Par exemple, le calcul d'une similarité Cosinus ne donne pas le même résultat qu'un test d'hypothèse.

Le tableau 5.6 récapitule les méthodes statistiques utilisées pour la comparaison d'attributs. Pour chaque méthode, nous précisons le type de résultat obtenu à interpréter. Nous distinguons trois types : les *p-values* pour les tests d'hypothèse, les coefficients de corrélations notés c dont les valeurs sont comprises entre -1 et 1, et les ratios notés r dont les valeurs sont comprises entre 0 et 1.

Méthode statistique	***p*-value**	**Corrélation** $c \in [-1, 1]$	**Ratio** $r \in [0, 1]$
Distance et ratio de Levenshtein			x
Coefficient de superposition			x
Test de Kolmogorov-Smirnov	x		
Test d'indépendance du Khi-deux	x		
Similarité Cosinus		x	
Rho de Spearman		x	
Indice binaire de Jaccard			x
Test de Kruskal-Wallis	x		
Corrélation bisériale ponctuelle		x	

TABLE 5.6 – Critère utilisé pour l'interprétation du résultat d'une méthode statistique

Cependant, nous souhaitons fournir à l'utilisateur une aide à la décision qui se base sur les trois indicateurs, et nous désirons le faire de façon visuelle et facile à comprendre. De ce fait, nous introduisons un code couleur. Pour chaque critère, si le résultat obtenu indique une bonne liaison entre les deux attributs, alors le critère est affiché en vert ; si le résultat semble indiquer une liaison mais sans grande confiance, alors le critère se colore en orange ; si le résultat n'indique pas de liaison, alors le critère apparaît en rouge. Ce mode de fonctionnement nous permet aussi de ne pas renvoyer la valeur exacte du critère, qu'un utilisateur qui n'est pas expert en la matière pourrait avoir du mal à interpréter.

Le tableau 5.7 présente la couleur renvoyée à l'utilisateur selon la valeur du critère, et plus particulièrement de la méthode employée pour évaluer le critère. Notons que pour le troisième critère, dans le cas où aucune comparaison statistique n'est effectuée car les natures des attributs ne sont pas compatibles, nous en informons aussi l'utilisateur en indiquant « inconnu » (en gris). Pour le coefficient de corrélation, nous utilisons sa valeur absolue notée $|c|$.

Apportons quelques précisions concernant les tests d'hypothèse. Pour les tests de Kolmogorov-Smirnov, d'ajustement du Khi-deux et de Kruskal-Wallis, l'hypothèse nulle formulée est celle de la similitude des attributs comparés. À l'inverse, le test d'indépendance du Khi-deux, comme son nom l'indique, formule comme hypothèse nulle l'indépendance des deux attributs comparés. Ainsi, la *p-value* doit être interprétée de manière différente. Par exemple, dans le cadre du test d'indépendance du Khi-deux, l'hypothèse H0 formulée est celle de l'indépendance des deux attributs comparés. Ainsi, lorsque la *p-value* est inférieure à 5%, nous rejetons l'hypothèse nulle, c'est-à-dire que nous rejetons l'indépendance des attributs : les deux attributs sont donc considérés comme liés, et le critère s'affiche en vert. À l'inverse, lorsque la *p-value* est supérieure à 10%, nous ne pouvons pas rejeter l'hypothèse d'indépendance, ce qui a pour résultat d'afficher le critère en rouge. Entre ces deux cas, c'est-à-dire lorsque la *p-value* est comprise entre 5% et 10%, nous rejetons toujours l'hypothèse nulle donc nous considérons les attributs comme liés, mais avec un niveau de confiance moins élevé, ce critère est alors affiché en orange (liaison modérée).

Indicateur	**p-value** (similitude)	**p-value** (indépendance)	**Corrélation** $c \in [-1, 1]$	**Ratio** $r \in [0, 1]$
Bonne liaison	$0,1 < p$	$p < 0,05$	$0,8 < \|c\|$	$0,8 < r$
Liaison moyenne	$0,05 < p < 0,1$	$0,05 < p < 0,1$	$0,5 < \|c\| < 0,8$	$0,5 < r < 0,8$
Pas de lien	$p < 0,05$	$0,1 < p$	$\|c\| < 0,5$	$r < 0,5$

TABLE 5.7 – Code couleur renvoyé à l'utilisateur selon la valeur d'un critère

5.4.2 Algorithme de vérification manuelle

Nous proposons un premier algorithme pour l'assistance à l'utilisateur lors de la comparaison des attributs d'une nouvelle occurrence des données aux notions déjà existantes dans la référence adéquate. Cet algorithme est utilisé lorsque l'utilisateur fait manuellement les associations entre les attributs de la nouvelle occurrence des données et la notion qui semble correspondre, et souhaite vérifier la force de cette association à travers les différents critères disponibles. Pour rappel, la comparaison s'effectue alors entre le nouvel attribut et l'attribut principal qui représente la notion.

Cet algorithme est plutôt utilisé dans le cas où l'utilisateur connaît les données et peut lier notions et nouveaux attributs très facilement, la vérification ne servant ici qu'à assurer qu'il n'y ait pas d'erreur ou de surprise dans les nouvelles données. En revanche, l'algorithme devient peu utile dans le cas où l'utilisateur n'a pas de connaissance appro-

fondie des nouvelles données, et/ou qu'il y a un grand nombre de notions et d'attributs à lier, ce qui rendrait le travail long et fastidieux.

L'algorithme 1 présente, en pseudo-code, l'algorithme que nous avons déployé pour cet usage. Le nouveau fichier f possède des attributs notés $\{\alpha_{i,new}\}_{i\in\mathbb{N}^*}$, chaque $\alpha_{i,new}$ formant un couple avec une notion ν_i ; le couple $(\alpha_{i,new}, \nu_i)$ a été saisi par l'utilisateur. Rappelons que pour la comparaison, la notion ν_i est représentée par son attribut principal, mais nous utilisons le terme notion pour simplifier. Pour chaque couple $(\alpha_{i,new}, \nu_i)$, les trois critères, à savoir le ratio de Levenshtein sur l'intitulé des attributs, le coefficient de superposition et une méthode statistique, sont respectivement notés lev_i, ovl_i et sta_i. Précisons que les fonctions `levenshtein`, `overlap` et `statistic` renvoient l'indication destinée à l'utilisateur comme présenté dans le tableau 5.7, et non les valeurs exactes des comparaisons effectuées.

Algorithme 1 : Algorithme de vérification manuelle

Données : $(\alpha_{i,new}, \nu_i)_{i\in\mathbb{N}^*}$
$lev_i \leftarrow 0$;
$ovl_i \leftarrow 0$;
$sta_i \leftarrow 0$;
pour chaque $(\alpha_{i,new}, \nu_i)_{i\in\mathbb{N}^*}$ **faire**
  $lev_i \leftarrow levenshtein(\alpha_{i,new}, \nu_i)$;
  $ovl_i \leftarrow overlap(\alpha_{i,new}, \nu_i)$;
  $sta_i \leftarrow statistic(\alpha_{i,new}, \nu_i)$;
fin
retourner $(lev_i, ovl_i, sta_i)_{i\in\mathbb{N}^*}$;

La figure 5.2 montre comment l'utilisateur peut se servir de cet algorithme dans l'interface de HOUDAL. Pour chaque attribut de la nouvelle occurrence des données, après avoir choisi la référence adéquate, l'utilisateur sélectionne à l'aide du menu déroulant la notion qui lui semble la mieux adaptée. Nous voyons sur la partie gauche de la figure que pour l'attribut `CD_PATRIM`, l'utilisateur a sélectionné la notion Code patrimoine. L'infobulle à côté de l'intitulé de l'attribut donne quelques précisions sur celui-ci : nature de l'attribut, nombre de valeurs distinctes et nombre de valeurs nulles. Enfin, l'icône bleue en forme d'œil permet de donner un aperçu des données de l'attribut en donnant quelques valeurs distinctes.

La partie droite de la figure montre comment est utilisé le code couleur introduit dans le tableau 5.7. Dans cet exemple, le premier critère, à savoir la comparaison des intitulés d'attribut, indique qu'il n'y a pas de liaison entre les deux attributs comparés. En revanche, le coefficient de superposition et le critère statistique indiquent une bonne liaison. Ceci pourrait laisser croire que l'attribut principal de la notion est bien lié au nouvel attribut en termes de valeurs, mais que leurs intitulés sont différents. Enfin, chaque critère dispose d'une infobulle qui permet d'indiquer à l'utilisateur la méthode statistique utilisée.

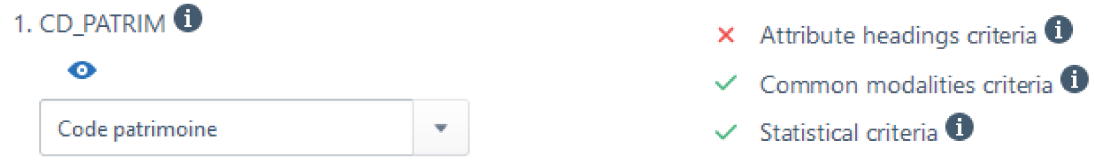

FIGURE 5.2 – Utilisation de l'algorithme manuel dans l'interface utilisateur

5.4.3 Algorithme de suggestion automatique

Pour pallier les limites d'utilisation du premier algorithme d'assistance, nous proposons une seconde option à l'utilisateur, plus orientée vers la suggestion. Ici, nous privilégions le cas de figure où l'utilisateur ne connaît pas la nouvelle occurrence des données, et/ou il y a trop de notions et d'attributs à lier et le faire à la main serait extrêmement chronophage. Nous proposons alors, pour chaque attribut, de le confronter à toutes les notions (représentées par leur attribut principal) existantes dans la référence, pour ne sélectionner que le meilleur couple attribut - notion trouvé grâce aux différents critères de comparaison.

L'avantage de cet algorithme est qu'il est utilisable dans tous les cas. En effet, même si l'utilisateur connaît bien ses données et s'il y a peu de notions et de nouveaux attributs à connecter, il est toujours préférable qu'un algorithme se charge de détecter les liaisons évidentes, quitte à ce que l'utilisateur doive corriger certains résultats erronés. Ainsi, nous pensons qu'il est pertinent d'avoir systématiquement recours à cette suggestion automatique, afin d'améliorer au mieux l'expérience utilisateur au sein du lac de données. Rajoutons aussi que ceci n'est en définitive qu'une aide à la décision, et que c'est toujours l'utilisateur qui doit valider en définitive.

L'algorithme 2 présente, en pseudo-code, l'algorithme que nous avons déployé pour cet usage. Nous réutilisons des notations similaires que celles utilisées dans l'algorithme précédent : le nouveau fichier de données f possède des attributs notées $\{\alpha_{i,new}\}_{i\in\mathbb{N}^*}$, et l'ensemble des notions de la référence concernée est noté $\{\nu_j\}_{j\in\mathbb{N}^*}$. Là aussi, pour la comparaison, la notion ν_j est représentée par son attribut principal, mais nous utilisons le terme notion pour simplifier.

Pour chaque nouvel attribut $\alpha_{i,new}$, nous le comparons à chaque notion ν_j, donc nous calculons les trois critères pour chaque couple $(\alpha_{i,new}, \nu_j)_{i\in\mathbb{N}^*,j\in\mathbb{N}^*}$. Ensuite, nous utilisons une heuristique simple pour déterminer le « meilleur » couple $(\alpha_{i,new}, \nu_j)$ en nous basant sur le tableau 5.7. Concrètement, si un critère indique une bonne similarité (vert), nous comptons 2 points ; si un critère donne une similarité modérée (orange), nous comptons 1 point ; si le critère indique qu'il n'y a pas de similarité, c'est 0 point. Nous effectuons ensuite la somme des points pour obtenir un score, et nous sélectionnons la notion ayant le meilleur score. Si plusieurs notions ont le même score, alors nous priorisons les critères : le troisième (méthode statistique) est le plus important, suivi du second (valeurs en commun), pour terminer par le premier (proximité des intitulés d'attributs). S'il reste encore une égalité après cela, nous sélectionnons de manière aléatoire, en laissant le soin à l'utilisateur de vérifier. La notion qui présente la meilleure similarité avec l'attribut

$\alpha_{i,new}$ est notée ν_i^{max}. Comme pour l'algorithme précédent, l'utilisateur n'obtient pas les valeurs exactes des critères mais seulement l'indication qui lui est destinée (tableau 5.7).

Algorithme 2 : Algorithme de vérification automatique

Données : $(\alpha_{i,new}, \nu_j)_{i\in\mathbb{N}^*, j\in\mathbb{N}^*}$
pour chaque $(\alpha_{i,new})_{i\in\mathbb{N}^*}$ **faire**
  $lev_{ij} \leftarrow 0$;
  $ovl_{ij} \leftarrow 0$;
  $sta_{ij} \leftarrow 0$;
  pour chaque $(\nu_j)_{j\in\mathbb{N}^*}$ **faire**
    $lev_{ij} \leftarrow levenshtein(\alpha_{i,new}, \nu_j)$;
    $ovl_{ij} \leftarrow overlap(\alpha_{i,new}, \nu_j)$;
    $sta_{ij} \leftarrow statictic(\alpha_{i,new}, \nu_j)$;
  fin
  $(\nu_i^{max}, lev_i^{max}, ovl_i^{max}, sta_i^{max}) \leftarrow heuristic(lev_{ij}, ovl_{ij}, sta_{ij})$;
fin
retourner $(\nu_i^{max}, lev_i^{max}, ovl_i^{max}, sta_i^{max})_{i\in\mathbb{N}^*}$;

5.4.4 Discussion

Le premier problème que nous rencontrons est posé par le deuxième algorithme (suggestion automatique) et concerne sa complexité, et par conséquent le temps de calcul. En effet, nous calculons les trois critères pour chaque couple notion - nouvel attribut possible. Ainsi, avec n notions et p nouveaux attributs, la complexité de cet algorithme est $O(n*p)$, ce qui peut rapidement devenir très lourd à calculer lorsque le nombre de notions et d'attributs augmente. D'autant plus que le coefficient de superposition et la plupart des méthodes statistiques nécessitent de parcourir toutes les données des attributs : si les fichiers contiennent beaucoup d'enregistrements, ces calculs peuvent eux aussi devenir très longs.

Pour réduire les temps de calculs, nous voulons mettre en place un processus de sélection, afin de ne calculer les critères que pour les couples notion - attribut qui semblent suffisamment pertinents. Nous pourrions, par exemple, nous baser sur la nature des attributs afin d'écarter en amont certaines comparaisons qui ne nous semblent pas pertinentes, et ainsi gagner du temps en optimisant le temps de calcul.

Un autre problème concernant la suggestion automatique est l'heuristique utilisée pour sélectionner la notion la plus adaptée à chaque nouvel attribut. En effet, même si les critères utilisés pour comparer des attributs entre eux sont pertinents, et que nous avons obtenus des résultats concluants dans notre cas d'usage, utiliser comme heuristique le fait de simplement donner un score par attribut valant 0, 1 ou 2 puis additionner ces scores est une méthode assez naïve. Nous considérons donc que cette approche mériterait d'être revue dans des travaux futurs. Nous pouvons imaginer laisser la possibilité à l'utilisateur de donner plus ou moins d'importance à certains critères en fonction de ses

connaissances sur les données. Si par exemple, l'utilisateur sait que les intitulés d'attributs sont strictement identiques, il est alors possible d'affecter un poids plus conséquent à ce critère.

5.5 Modélisation des métadonnées avec goldMEDAL

QSTR aide l'utilisateur à créer de nouvelles métadonnées pour les données structurées générées de manière périodique, et ceci en effectuant des comparaisons entre attributs issues de fichiers différents. Nous souhaitons désormais transformer ces nouvelles informations en métadonnées selon les concepts du métamodèle goldMEDAL, dans l'optique de pouvoir les enregistrer au sein du système de gestion des métadonnées du lac de données HOUDAL.

Toutefois, HOUDAL contient déjà des métadonnées sur les données présentes en son sein. Nous devons donc être vigilant à ne pas perturber l'organisation des métadonnées déjà en place, car elles sont toujours nécessaires au bon fonctionnement du lac, malgré le nouveau besoin énoncé dans ce chapitre. En outre, les nouvelles métadonnées que nous souhaitons enregistrer ne sont pas au même niveau de granularité que les métadonnées existantes : pour les données structurées, elles décrivent le fichier de données dans son intégralité, tandis que les nouvelles informations concernent les attributs des données, un niveau de granularité plus fin. Puisqu'un fichier de données structurées peut contenir plusieurs attributs, la question se pose de savoir où placer le curseur concernant le niveau de granularité, et déterminer si une entité de données désigne un attribut, un fichier, ou potentiellement les deux. Nous avons montré dans le chapitre 3 que goldMEDAL permet de gérer des niveaux de granularité multiples par le biais de groupements, mais il nous reste à étudier comment gérer le cas où des entités de données ne possèdent pas la même granularité.

Nous commençons par présenter dans la section 5.5.1 comment les informations générées par QSTR peuvent devenir des métadonnées et être modélisées par goldMEDAL. Ensuite, la section 5.5.2 illustre comment goldMEDAL peut contenir des métadonnées à deux niveaux de granularité, c'est-à-dire les métadonnées déjà existantes au niveau « fichier », et les nouvelles métadonnées au niveau « attribut ».

5.5.1 Instanciation du métamodèle pour les attributs

Pour enregistrer les métadonnées portant sur les attributs des données structurées, nous utilisons notre métamodèle de métadonnées goldMEDAL. Nous instancions donc le métamodèle conceptuel en reprenant les quatre concepts de goldMEDAL, à savoir entité de données, groupement, lien et processus, et nous montrons comment ils modélisent les métadonnées liées aux attributs générées par QSTR. Les exemples sont ensuite déclinés au niveau logique en reprenant la théorie des graphes. Nous reprenons dans cette section les mêmes notations que nous avons utilisées dans le chapitre 3.

5.5.1.1 Entité de données

Un attribut est modélisé comme une entité de données. Chaque entité peut avoir des propriétés, comme par exemple le nombre de valeurs distinctes (modalités), le nombre de valeurs nulles, etc.

Au niveau logique, une entité de données est représentée par un nœud porteur d'attributs.

Exemple 14 *Soit un fichier f ayant trois attributs. En enregistrant le fichier dans le lac de données, trois nœuds n_1, n_2 et n_3 sont créés pour modéliser les trois entités de données correspondantes.*

Rappelons que le niveau de granularité d'une entité de données est flexible, et s'adapte aux besoins de l'utilisateur. Ici, la granularité de l'entité de données est au niveau de l'attribut, contrairement à ce qui a été présenté dans le chapitre 4 où ce sont les fichiers qui sont modélisés comme des entités de données. Dans l'exemple ci-dessus, nous ne précisons pas comment le fichier en tant que tel est modélisé dans les métadonnées, car le niveau de granularité n'est pas le même. La gestion de cette différence de granularité est détaillée dans la section 5.5.2.

5.5.1.2 Groupement

Une entité de données peut appartenir à plusieurs groupements. Un groupement est un ensemble de groupes, et un groupe rassemble des entités de données selon des propriétés communes. Par exemple, la nature des attributs peut être un groupement, constitué de cinq groupes (quantitatif, catégoriel, textuel, booléen, date). L'ensemble des notions d'une référence peut constituer un autre groupement, chaque notion étant un groupe. Ainsi, les attributs qui sont liés à une notion sont les entités de données rassemblées dans le groupe correspondant.

Au niveau logique, un groupe est modélisé par une hyperarête (c'est-à-dire une arête pouvant lier plus de deux nœuds), un groupement est donc un ensemble d'hyperarêtes.

Exemple 15 *Soit $G_1 = \{\Gamma_{11}, \Gamma_{12}, \Gamma_{13}, \Gamma_{14}, \Gamma_{15}\}$ un groupement représentant les différentes natures possibles d'un attribut. Les groupes Γ_{11}, Γ_{12}, Γ_{13}, Γ_{14} et Γ_{15} désignent respectivement les natures quantitatif, catégoriel, textuel, booléen et date.*

Au niveau logique, le groupement est traduit en un ensemble d'hyperarêtes noté $H_1 = \{\theta_{11}, \theta_{12}, \theta_{13}, \theta_{14}, \theta_{15}\}$, où les hyperarêtes traduisent les groupes.

Représentons le niveau logique de l'exemple 15 avec la figure 5.3. Les nœuds n_1 et n_3 sont contenus dans θ_{11}, ce qui signifie que ces deux attributs sont quantitatifs. n_2 est un attribut catégoriel, c'est pourquoi il se retrouve dans θ_{12}.

5.5.1.3 Lien

Il existe deux types de liens : ceux reliants des entités de données et les liens hiérarchiques entre groupes d'un groupement. En ce qui concerne les liens entre entités de

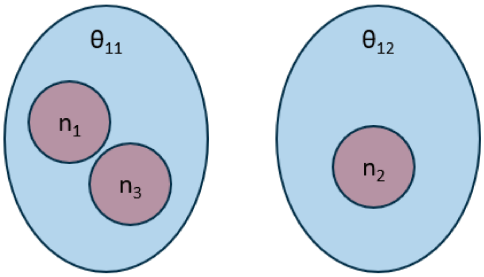

FIGURE 5.3 – Groupement de goldMEDAL appliqué aux attributs

données, nous pouvons enregistrer le résultat d'une comparaison menée entre deux attributs en tant que liaison de similarité, afin de pouvoir fournir le résultat plus vite lors d'une potentielle future comparaison.

Au niveau logique, ce lien est traduit par une arête non orientée qui connecte deux nœuds ensemble.

Exemple 16 *Soit e_1 et e_2 deux entités de données représentant des attributs. Une comparaison entre ces deux entités mène à la création d'un lien l_1 : $e_1 \underset{l_1}{\longrightarrow} e_2$. Notons que l_1 est non orienté, donc nous pouvons aussi écrire $e_2 \underset{l_1}{\longrightarrow} e_1$.*

Au niveau logique, les entités de données sont traduites par deux nœuds n_1 et n_2 sont reliés par une arête $a_1 = (n_1, n_2)$ qui traduit le lien.

L'exemple 16 est représenté graphiquement par la figure 5.4.

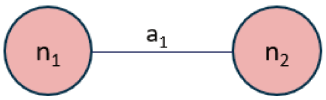

FIGURE 5.4 – Lien entre entités de goldMEDAL appliqué aux attributs

Nous pouvons aussi créer des liens hiérarchiques entre groupes. Si l'ensemble des références est un groupement, il est alors possible de le lier à celui des notions par un lien hiérarchique, puisqu'une référence contient plusieurs notions. C'est pourquoi chaque notion (groupe) est connectée à une référence (groupe) par un lien hiérarchique.

Ce type de lien se traduit au niveau logique par une arête orientée reliant deux hyperarêtes.

Exemple 17 *Soit $G_2 = \{\Gamma_{21}\}$ l'ensemble des références avec Γ_{21} une référence et $G_3 = \{\Gamma_{31}, \Gamma_{32}, \Gamma_{33}\}$ l'ensemble des notions avec Γ_{31}, Γ_{32} et Γ_{33} trois notions. Le lien hiérarchique l_2 relie les notions à la référence : $\Gamma_{31} \underset{l_2}{\longrightarrow} \Gamma_{21}, \Gamma_{32} \underset{l_2}{\longrightarrow} \Gamma_{21}, \Gamma_{33} \underset{l_2}{\longrightarrow} \Gamma_{21}$. Inversement, $\Gamma_{21} \underset{l_2^{-1}}{\longrightarrow} \{\Gamma_{31}, \Gamma_{32}, \Gamma_{33}\}$.*

Traduisons maintenant cela au niveau logique. L'arête a_2 matérialise le lien hiérarchique entre H_2 et H_3 : $\theta_{31} \underset{a_2}{\longrightarrow} \theta_{21}$, $\theta_{32} \underset{a_2}{\longrightarrow} \theta_{21}$ et $\theta_{33} \underset{a_2}{\longrightarrow} \theta_{21}$. Aussi, nous avons $\theta_{21} \underset{a_2^{-1}}{\longrightarrow} \{\theta_{31}, \theta_{32}, \theta_{33}\}$.

La figure 5.5 illustre l'exemple 17.

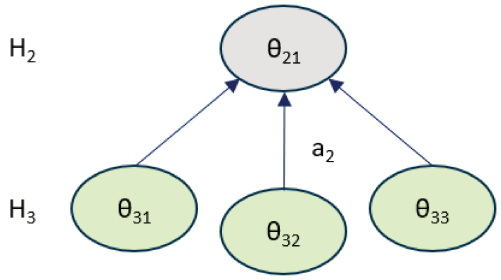

FIGURE 5.5 – Lien hiérarchique de goldMEDAL appliqué aux attributs

5.5.1.4 Processus

Enfin, les attributs peuvent être liés entre eux à travers des processus. Un processus peut matérialiser l'évolution d'un attribut au fil du temps à travers les nouvelles occurrences d'un fichier. Par exemple, les attributs au sein d'une notion peuvent être connectés par une série de processus illustrant les arrivées successives de nouvelles occurrences des données dans le lac. Dans ce cas, l'attribut le plus récent de la notion est en entrée de ce processus, tandis que le nouvel attribut est alors en sortie de celui-ci.

Notons que dans ce cas de figure précis, le concept de processus ne désigne pas un « processus » de traitement des données à proprement parler, mais simplement une évolution de l'attribut au fil des occurrences de données. De plus, nous pourrions aussi modéliser ces évolutions avec un simple lien orienté entre entités de données. Nous faisons toutefois le choix d'utiliser le processus pour modéliser cette information, pour nous laisser la possibilité de prendre en compte dans le futur de nouvelles possibilités concernant les notions et les attributs comme évoqué dans la section 5.2.3. Nous pourrions alors modéliser aisément le fait qu'un attribut se sépare en plusieurs attributs dans une nouvelle occurrence de données, ou que plusieurs attributs fusionnent en un seul, puisqu'un processus peut rassembler plusieurs entités de données en entrée et en sortie de celui-ci.

Au niveau logique, un processus est matérialisé par une hyperarête orientée reliant des nœuds. Notons que dans le cas de figure que nous présentons ici, vu qu'il n'y a qu'un nœud en entrée et un en sortie, une simple arête orientée suffirait.

Exemple 18 *Soit e_1, e_2 et e_3 trois entités de données. Le processus $P_1 = \{I_1, O_1\}$ matérialise l'évolution de e_1 à e_2, alors $I_1 = \{e_1\}$ et $O_1 = \{e_2\}$. De même, $P_2 = \{I_2, O_2\}$, matérialise l'évolution de e_2 à e_3, donc nous avons $I_2 = O_1 = \{e_2\}$ et $O_2 = \{e_3\}$.*

Au niveau logique, nous notons $\Pi_1 = \{\Upsilon_1, \Omega_1\}$, avec $\Upsilon_1 = \{n_1\}$ et $\Omega_1 = \{n_2\}$. De même, nous avons $\Pi_2 = \{\Upsilon_2, \Omega_2\}$, avec $\Upsilon_2 = \Omega_1 = \{n_2\}$ et $\Omega_2 = \{n_3\}$.

L'exemple 18 est décrit de manière visuelle à travers la figure 5.6. Ici, il est intéressant de noter que les entrées et sorties de chaque processus sont des singletons, c'est-à-dire un seul attribut à la fois. De plus, pour chaque processus P_n, l'entrée de P_{n+1} est égale à la sortie de P_n.

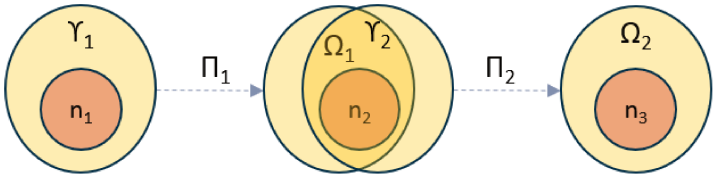

FIGURE 5.6 – Processus de goldMEDAL appliqué aux attributs

5.5.2 goldMEDAL à deux niveaux de granularité

Dans l'implémentation d'un lac de données HOUDAL, comme nous nous basons sur goldMEDAL pour le système de gestion des métadonnées, le niveau de granularité des entités de données est modulable selon les besoins de l'utilisateur. Généralement, une entité de données correspond à un fichier (ou une table d'une base de données), et c'est ce que nous avions initialement décidé pour notre lac de données. Néanmoins, à la lumière de ce travail sur les attributs, nous sommes obligés de revoir le niveau de granularité en ce qui concerne les données structurées, tout en évitant de perdre les métadonnées déjà existantes.

Nous étudions ici comment nous pouvons gérer ce problème de granularité différente dans goldMEDAL, en proposant deux solutions. La première est de faire cohabiter des entités de données n'ayant pas la même granularité, et la seconde consiste à considérer les niveaux de granularité supérieurs comme des groupes. Nous terminons en discutant de ces deux approches et des pistes d'évolution.

5.5.2.1 Cohabitation des fichiers et attributs

La première solution que nous proposons pour avoir des métadonnées à granularité différente dans goldMEDAL est d'étiqueter les entités de données selon leur niveau de granularité. Il est en effet tout à fait possible d'indiquer le niveau de granularité dans les propriétés de l'entité de données. Ainsi, nous créons pour chaque attribut une nouvelle entité de données, qui possède ses propres propriétés. Ces nouvelles entités de données cohabitent donc avec celles qui étaient déjà présentes dans le système de métadonnées du lac. Pour connecter une entité de données « attribut » à une entité « fichier », nous utilisons le concept de lien. Puisque dans goldMEDAL, un lien entre deux entités de données peut être orienté ou non, nous créons un lien orienté du nœud représentant l'attribut vers le nœud représentant le fichier, pour signifier l'appartenance d'un attribut à un fichier. Ainsi, si un fichier possède trois attributs, nous avons au total quatre entités de données : une pour chaque attribut, et une pour le fichier.

Exemple 19 *Soit e_1, e_2 et e_3 trois entités de données représentant des attributs, et e_f l'entité de données qui modélise le fichier auquel elles appartiennent. Le lien l_2 permet de connecter les attributs au fichier : $e_1 \xrightarrow[l_2]{} e_f$, $e_2 \xrightarrow[l_2]{} e_f$ et $e_3 \xrightarrow[l_2]{} e_f$. En notation fonctionnelle, $l_2(e_1) = e_f$, $l_2(e_2) = e_f$ et $l_2(e_3) = e_f$. Inversement, $e_f \xrightarrow[l_2^{-1}]{} \{e_1, e_2, e_3\}$.*

Au niveau logique, les trois nœuds n_1, n_2 et n_3 sont reliés au nœud n_f par une arête orientée : $a_2 = (n_1, n_f) = (n_2, n_f) = (n_3, n_f)$.

Nous illustrons l'exemple 19 à travers la figure 5.7. Les trois nœuds n_1, n_2 et n_3 représentant des attributs sont chacun relié par l'arête a_2 au nœud n_f, qui modélise le fichier auquel elles appartiennent.

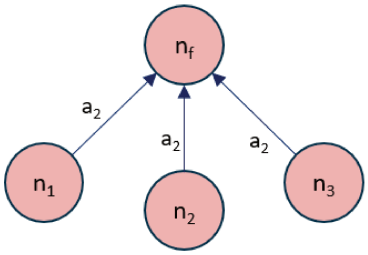

FIGURE 5.7 – Cohabitation d'entités de données aux granularités différentes

Avec cette approche, nous pouvons utiliser les autres concepts de goldMEDAL (groupement, lien, processus) pour les entités de données « attribut », comme nous l'avons présenté dans la section 5.5.1. En plus de cela, nous pouvons aussi utiliser ces concepts au niveau des entités de données « fichier » aussi, puisque nous en avons toujours besoin. Cette solution est donc intéressante car reprend exactement les concepts de goldMEDAL tels que nous les avons proposés, ce qui tend à prouver sa généricité et sa capacité à s'adapter à toutes sortes de contraintes.

En revanche, avec cette solution, il nécessaire de bien documenter et expliquer cette différence de granularité au sein des entités de données. Il est en effet important que l'utilisateur, lors de son exploration des métadonnées du lac, ne soit pas confus à cause de ce fonctionnement. D'autre part, nous n'exploitons pas vraiment les opportunités offertes par la différence de niveau de granularité, ce qui pourrait s'avérer intéressant sur plusieurs aspects. Nous discutons de cela à la fin de cette section.

5.5.2.2 Modification du niveau de granularité

Une seconde solution consiste à changer le niveau de granularité des entités de données présentes dans le lac : ainsi, une entité ne traduit plus un fichier mais un attribut. En faisant ainsi, nous devons créer un nouveau groupement « fichier » qui indique l'appartenance d'un attribut à un certain fichier. Ce groupement est en fait l'ensemble des entités de données avant cette modification effectuée pour ajouter les métadonnées sur les attributs.

Exemple 20 *Soit e_1, e_2 deux attributs appartenant à un fichier et e_3, e_4 deux autres attributs contenus dans un autre fichier. Nous notons le groupement des fichiers G_f, et ses groupes Γ_{f1} et Γ_{f2} représentent les deux fichiers. Ainsi, nous avons $\Gamma_{f1} = \{e_1, e_2\}$ et $\Gamma_{f2} = \{e_3, e_4\}$.*

Au niveau logique, l'ensemble d'hyperarêtes est noté $H_f = \{\theta_{f1}, \theta_{f2}\}$, et les nœuds n_1 et n_2 (resp. n_3 et n_4) sont contenus dans θ_{f1} (resp. θ_{f2}).

La figure 5.8 illustre l'exemple 20. Les nœuds n_1 et n_2 sont rassemblés dans θ_{f1}, et n_3 et n_4 appartiennent à θ_{f2}.

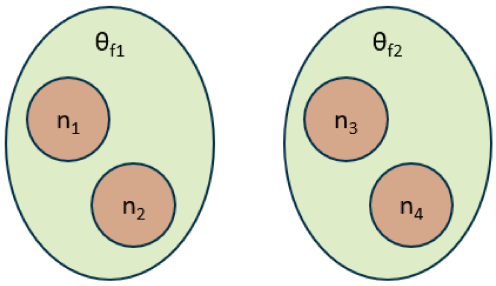

FIGURE 5.8 – Fichiers de données transformés en groupes

Toutefois, puisque les fichiers étaient précédemment les entités de données, ils peuvent être rassemblés dans des groupes, ou connectés entre eux à travers des liens et des processus. Mais avec ce changement de granularité, ce sont désormais des groupes d'un groupement, et parmi les concepts de goldMEDAL, seuls les liens hiérarchiques permettent de connecter des groupes entre eux. Ainsi, il n'est pas possible relier les groupes « fichier » entre eux avec des liens (de similarité par exemple) ou des processus. Cependant, malgré le besoin métier ayant motivé ce chapitre, nous avons tout de même besoin de conserver ces connexions entre fichiers, ce qui nous bloque ici. En adoptant cette solution, il nous faudrait alors apporter des modifications à notre métamodèle goldMEDAL.

Néanmoins, cette approche de modifier le niveau de granularité des fichiers apporte une vision intéressante, qui ouvre de nouvelles pistes de réflexion pour notre métamodèle de métadonnées. Nous discutons de cela ci-après.

5.5.2.3 Discussion : évolutions possibles de goldMEDAL

Les deux solutions proposées sont deux façons différentes de gérer des métadonnées à des niveaux de granularité différents au sein du métamodèle de métadonnées. Tandis que la première réutilise les concepts de goldMEDAL pour faire cohabiter des entités de données de granularité différente, la seconde nécessite d'apporter des modifications aux concepts du métamodèle, mais ouvre aussi la porte à de nouvelles opportunités.

Si nous considérons que les entités de données sont les attributs, les liens entre entités peuvent donner une vision plus fine d'un lien existant entre fichiers. Par exemple, si nous enregistrons plusieurs calculs de similarité entre attributs de deux fichiers différents, nous

pourrions alors générer un lien entre ces deux fichiers, qui serait constitué en agrégeant les liens existants au niveau de granularité plus fin, à savoir les attributs.

De même que pour les liens entre entités de données, les processus liant des fichiers se retrouvent à un niveau de granularité supérieur. Les processus entre entités de données (attributs) permettent de décrire l'évolution d'un fichier à un niveau de granularité plus fin, et donc de manière plus détaillée, en indiquant notamment comment les attributs évoluent.

Pour dresser un bilan de ces idées, nous pouvons suggérer que chaque concept de goldMEDAL (entité de données, groupement, lien, processus) puisse être à des niveaux de granularité différents. Pour les entité de données, cela est déjà prévu, nous l'avons illustré à travers la première solution en faisant cohabiter attributs et fichiers au sein des entités. De plus, les groupements peuvent être liés entre eux par des liens hiérarchiques. Toutefois, les liens entre entités et processus n'ont pas de notion de granularité dans goldMEDAL, et cela pourrait faire l'objet d'une évolution de ce dernier.

Avoir des liens et processus hiérarchiques permettrait de nouvelles analyses des métadonnées. En effet, selon les besoins de l'utilisateur, ce dernier peut explorer les métadonnées à un niveau de granularité fin comme les attributs, ou bien à plus gros grain, c'est-à-dire au niveau fichier, voire une collection de fichiers (ou une base de données). En ayant des métadonnées à chaque niveau de granularité, nous permettrions à l'utilisateur de naviguer à sa guise. Cette navigation entre les niveaux de granularité fait penser aux opérations OLAP de *roll-up* et *drill-down*.

Par ailleurs, le fait de pouvoir regrouper des liens et des processus pour monter à un niveau de granularité plus élevé s'avère être intéressant dans d'autres cas de figure que celui des attributs présents dans les données structurées. Par exemple, considérons un script qui traite plusieurs fichiers bruts de manière séquentielle pour les nettoyer, et ainsi créer des fichiers raffinés, plus exploitables. Nous pourrions alors créer un processus dont l'entrée est l'ensemble des fichiers bruts et la sortie est l'ensemble des fichiers traités. Toutefois, nous pourrions aussi créer un processus par fichier, puisque le script les traite de manière séquentielle. Nous nous retrouvons ici dans un autre cas où avoir un « processus hiérarchique » présente un intérêt réel.

5.6 Conclusion

Nous avons adressé dans ce chapitre la problématique de création de métadonnées pour les données structurées générées de manière périodique, et dont le schéma peut être amené à évoluer au fil du temps. Par exemple, dans une nouvelle occurrence des données, un attribut peut changer d'intitulé, le codage de ses données peut évoluer, ou bien un nouvel attribut peut faire son apparition. Dans un lac de données, ces évolutions de schéma ne posent pas de problème lors de l'ingestion des données, car elles sont enregistrées sans schéma prédéfini. Toutefois, l'utilisateur peut être amené pour un besoin d'analyse à interroger simultanément toutes les occurrences des données, et ces évolutions de schéma deviennent alors un problème bloquant, puisque le schéma des données est

défini à l'interrogation dans le lac.

Nous avons proposé du vocabulaire pour aborder cette problématique. Nous définissons une référence comme un ensemble de fichiers générés périodiquement, mais dont le schéma peut légèrement évoluer d'une occurrence à l'autre. Une référence contient des notions, où une notion rassemble plusieurs attributs et représente l'information qu'ils portent, et ce en dépit des évolutions qu'ils peuvent subir au fil des occurrences. Enfin, chaque notion possède un attribut principal, qui sert à « représenter » l'information contenue dans une notion lors des comparaisons.

Fort de ce vocabulaire, nous proposons ensuite trois critères permettant de comparer deux attributs à l'arrivée d'une nouvelle occurrence des données dans le lac. Nous comparons ici l'attribut principal d'une notion aux nouveaux attributs entrants. Le premier critère calcule la proximité des intitulés des attributs. Dans un second temps, nous étudions le taux de valeurs communes entre les données des deux attributs. Le troisième critère est une méthode statistique qui dépend de la nature des deux attributs comparés.

En outre, nous proposons QSTR, un assistant à la comparaison des attributs et à la création de métadonnées. QSTR propose deux algorithmes d'aide à la décision pour l'utilisateur. Le premier consiste à vérifier manuellement les couples notion - nouvel attribut saisis manuellement par l'utilisateur, tandis que le second vise à détecter de manière automatique les notions et nouveaux attributs qui sont les plus proches, afin de les suggérer à l'utilisateur.

Enfin, nous avons ajouté ces nouvelles informations sur les attributs aux métadonnées existantes dans le lac de données. À ce titre, nous sommes obligés de jongler entre deux niveaux de granularité d'entités de données, puisque nous avons à la fois des informations sur les « fichiers » mais aussi sur les attributs. Nous avons donc illustré qu'il est possible de tout gérer avec goldMEDAL, soit en faisant cohabiter des entités de données qui n'ont pas la même granularité, soit en transformant les fichiers en groupes.

Chapitre 6

Conclusion générale

Sommaire

"Quietly yearning for what you don't have, while dreading losing what you do. For 99.9% of your race, that is the definition of reality. Desire and fear, baby."

The Analyst, *The Matrix Resurrections (2021)*

6.1 Bilan des contributions

Tous les travaux présentés dans ce manuscrit de thèse ont été motivés par une problématique métier rencontrée chez BIAL-X, et plus particulièrement lors des projets menés en collaboration avec les clients de l'entreprise. Cette problématique est le besoin de stocker, organiser et exploiter des mégadonnées, avec une forte coloration pour le domaine métier de l'habitat social, secteur métier clé pour l'activité économique de BIAL-X.

En réponse à cette problématique métier, nous avons proposé d'utiliser un lac de données, ce qui a induit des questions de recherche autour de la conception, la modélisation et l'implémentation d'un lac de données et son système de gestion des métadonnées, dans le but de stocker, organiser et exploiter des données hétérogènes. Nous pouvons catégoriser nos contributions de deux manières : celles sur la modélisation des métadonnées (section 6.1.1), et celles sur la mise en œuvre d'un lac de données (section 6.1.2). Nous terminons cette section de bilan en montrant comment nos contributions apportent aussi des réponses aux problèmes métiers rencontrés chez BIAL-X, et plus particulièrement pour les bailleurs sociaux avec qui l'entreprise travaille en étroite collaboration (section 6.1.3).

6.1.1 Contributions à la modélisation des métadonnées

Le système de gestion des métadonnées est un élément vital au bon fonctionnement du lac de données, et à ce titre, plusieurs équipes de recherche ont proposé leur approche. Toutefois, la plupart des propositions sont des implémentations dédiées à des besoins spécifiques, et difficilement réutilisables. Il existe en revanche des modèles de métadonnées qui se veulent plus génériques et explicites, donc réutilisables. Une étude comparative menée par nos soins démontre qu'aucun système ou modèle n'est totalement satisfaisant en termes de généricité ou de fonctionnalités supportées. C'est pourquoi nous avons proposé un modèle de métadonnées, puis son évolution en un métamodèle de métadonnées.

Le premier modèle de métadonnées, baptisé MEDAL, s'appuie sur une typologie de métadonnées intra-objet, inter-objets et globales, où un objet représente un ensemble homogène de données. La modélisation logique de MEDAL est basée sur la théorie des graphes. La motivation pour proposer MEDAL provient de l'introduction de six fonctionnalités clés pour la gestion des métadonnées de lacs de données. En nous basant sur ces fonctionnalités, une comparaison entre MEDAL et les différents systèmes de métadonnées proposés dans la littérature a montré que notre modèle était le plus complet au moment de l'étude.

Le modèle MEDAL a été présenté dans un article de conférence nationale [Scholly et al., 2019] et dans un article en atelier international [Sawadogo et al., 2019b].

Après la publication des travaux sur MEDAL, plusieurs équipes de recherche ont proposé leur propre modèle de métadonnées, visant à plus de généricité. Ces travaux nous ont poussé à revoir les six fonctionnalités pour les adapter spécifiquement aux modèles de métadonnées, et non aux systèmes au sens large, pour obtenir huit caractéristiques de modélisation des métadonnées. À la lumière de ces travaux et aussi lors d'échanges

sur des projets différents au sein du laboratoire ERIC, nous avons proposé le métamodèle goldMEDAL, une évolution de MEDAL. Il se distingue de son prédécesseur en se positionnant à un niveau d'abstraction plus élevé. Pour évaluer goldMEDAL, nous avons montré qu'il est capable de modéliser les fonctionnalités de tous les autres modèles, tout en en proposant de nouvelles.

La présentation de goldMEDAL a été publiée sous la forme d'un article dans un atelier international qui est la référence en informatique décisionnelle [Scholly et al., 2021b].

6.1.2 Contributions à la mise en œuvre d'un lac de données

En parallèle des travaux de recherche sur la modélisation des métadonnées, qui font intervenir des problématiques plus théoriques, des chercheurs se sont aussi penchés sur la mise en œuvre concrète d'un lac de données, c'est-à-dire d'une part son architecture fonctionnelle, et d'autre part les outils permettant de l'implémenter. Ces derniers sont nombreux, donc le choix est vaste pour l'implémentation d'un lac. En revanche, en ce qui concerne les architectures de lac de données, il n'en existe qu'un nombre réduit, et toutes ont tendance à compartimenter les données du lac. Ceci a motivé des contributions de notre part, en proposant notre architecture de lac de données et une implémentation associée, cette dernière possédant un module spécifique pour faciliter l'ingestion de données structurées.

HOUDAL, l'implémentation d'un lac de données que nous proposons dans le contexte de l'habitat social, se base sur une interface web avec laquelle l'utilisateur interagit. Le système de gestion des métadonnées du lac est implémenté avec la base de données graphe Neo4j. Avec HOUDAL, l'utilisateur peut déposer des fichiers de tous types dans le lac, et créer les métadonnées associées. Il peut aussi consulter les métadonnées existantes, pour identifier les entités de données qui pourraient répondre à un besoin d'analyse, pour ensuite les récupérer le cas échéant. HOUDAL se base sur notre proposition d'architecture, basée sur quatre composants : l'ingestion, le stockage de données, le système de gestion des métadonnées, et l'exploitation des données. Cette architecture présente l'intérêt d'être cyclique, c'est-à-dire que des nouvelles données créées lors d'une exploitation de données présentes dans le lac sont elles aussi enregistrées dans le lac, en passant par la couche d'ingestion pour créer les métadonnées associées à ces nouvelles données.

Les spécifications de notre architecture ainsi qu'HOUDAL ont été acceptées pour publication sous la forme d'un article de conférence internationale [Scholly et al., 2021a].

QSTR est un assistant à la création de métadonnées lors de l'insertion de données structurées. Intégré à HOUDAL, il a pour objectif de faciliter la création de métadonnées, qui est un processus primordial, mais peut aussi s'avérer être fastidieux. En particulier, il arrive parfois que des données structurées soient générées de manière périodique, mais dont le schéma évolue à chaque nouvelle occurrence des données. Si l'utilisateur est amené à devoir interroger simultanément toutes les occurrences de données, alors les différences de schéma deviennent un problème. QSTR assiste l'utilisateur en comparant la nouvelle occurrence des données en cours d'ingestion aux précédentes occurrences de données déjà enregistrées dans le lac. Basé sur des méthodes statistiques, il suggère à

l'utilisateur de potentielles similarités entre les attributs des nouvelles données avec ceux des données déjà existantes dans le lac. L'utilisateur a toujours le dernier mot, et doit valider ou invalider les suggestions de QSTR. *In fine*, les informations générées à l'aide de l'assistant sont modélisées avec le métamodèle goldMEDAL et enregistrées dans le système de métadonnées de HOUDAL.

6.1.3 Solutions apportées aux problèmes métiers

En plus des contributions scientifiques sur la thématique des lacs de données, nos travaux apportent aussi des réponses aux problèmes métiers rencontrés chez BIAL-X qui collabore avec des bailleurs sociaux. Nous divisons les solutions en deux grandes parties : d'une part, l'implémentation du lac de données HOUDAL, et du métamodèle de métadonnées goldMEDAL ; d'autre part, l'assistant à la création de métadonnées pour données structurées QSTR.

6.1.3.1 Le lac de données et son système de métadonnées

L'implémentation du lac de données HOUDAL et du système de métadonnées basé sur le métamodèle goldMEDAL permet aux bailleurs sociaux d'enregistrer au sein d'un même périmètre toutes les données dont ils ont besoin pour faire des analyses. Ceci permet d'éviter d'avoir des données éparpillées sur différents serveurs, en local ou sur le *cloud*, dans des bases de données, des fichiers, etc. De plus, grâce au métamodèle goldMEDAL, les métadonnées sont suffisamment génériques pour s'adapter à tous les cas d'usages que les utilisateurs du lac pourraient rencontrer. Les utilisateurs du lac peuvent donc qualifier les données de manière homogène, et ceci constitue une condition nécessaire au bon fonctionnement du système dans la durée. Cela permet en effet d'éviter d'avoir une gouvernance de données incomplète, et donc des métadonnées hétérogènes à l'échelle de l'entreprise.

Par ailleurs, avoir le système de métadonnées intégré directement au lac s'avère être plus pertinent qu'avoir à gérer la gouvernance des données de manière dissociée des données. Il existe de tels outils, comme DataGalaxy [1], qui permettent de maintenir un catalogue de métadonnées décrivant les données et leurs usages. Toutefois, ce catalogue est saisi *a posteriori*, c'est-à-dire que les utilisateurs enregistrent leurs données et les transformations dans un premier temps, puis décrivent tout ce qu'ils ont fait dans le catalogue dans un second temps, généralement à la fin du projet, lorsque tout fonctionne. BIAL-X a déjà été amené à utiliser un tel outil chez des clients, et a constaté qu'en général, la bonne saisie des informations dans l'outil est respectée au début lorsque les acteurs du projet sont motivés, mais finissent par le délaisser rapidement car c'est une tâche très fastidieuse et pénible. À l'arrivée, le catalogue est délaissé et finit par ne plus être utilisé.

À l'inverse, avec HOUDAL et son mode de fonctionnement *describe-on-write*, aucune entité de données ne peut être enregistrée dans le lac de données sans qu'elle soit qualifiée, i.e. que des métadonnées soient saisies par l'utilisateur. Ceci force les utilisateurs

1. `https://www.datagalaxy.com/`

à saisir les métadonnées immédiatement et évite l'effet de « lassitude » d'avoir à saisir et maintenir les métadonnées *a posteriori*. Enfin, nous considérons que ce mode de fonctionnement est plus pertinent car l'utilisateur est plus à même de saisir des métadonnées dès l'enregistrement des données plutôt que de devoir se remémorer ce qui a été fait en saisissant les informations plus tard, parfois des semaines ou des mois après.

Un autre avantage majeur proposé par HOUDAL est la possibilité d'adresser de nouveaux besoins ponctuels d'exploitation de données et ce de manière assez simple. Les utilisateurs du lac peuvent se permettre de déposer de nouvelles données dans le lac, notamment des données externes glanées sur le web, dont ils ne connaissent pas forcément la qualité ni le contenu en détail. Avec les métadonnées du lac de données, il est possible d'indiquer que les données proviennent de l'extérieur, que leur qualité est inconnue et qu'elles sont là pour un besoin ponctuel. Les autres utilisateurs du lac, qui n'ont pas besoin de ces données, pourront alors les ignorer sans problème.

Le besoin ponctuel peut aussi être décrit via les métadonnées du lac. Si, pour répondre à un besoin d'analyse, les données externes brutes sont retravaillées et qu'une nouvelle entité de données est créée (contenant par exemple des données nettoyées), tout ceci est décrit dans les métadonnées. Dans ce cas, il existe désormais une nouvelle représentation des données externes que les autres utilisateurs du lac sont susceptibles de réutiliser pour d'autres besoins, puisqu'elles ont été retravaillées et que le processus de nettoyage des données est documenté. Avec ces informations, les données externes retravaillées et documentées deviennent une source de données sur laquelle les autres utilisateurs peuvent compter à l'avenir.

6.1.3.2 Assistance à la création de métadonnées pour données structurées

Toutes les opportunités apportées par l'implémentation du lac de données HOUDAL reposent sur un élément clé : la bonne qualification des données lors de leur insertion dans le lac. C'est justement là qu'intervient QSTR, un assistant à la création de métadonnées pour données structurées, car le métamodèle de métadonnées goldMEDAL est efficace seulement si les métadonnées ont été correctement saisies. Nous avons été amenés à constater de manière empirique que la saisie des métadonnées par l'utilisateur peut s'avérer être un processus fastidieux, voire même complexe dans certains cas. QSTR assiste l'utilisateur dans cette démarche pour la rendre la plus facile et la moins fastidieuse possible.

Plus spécifiquement, nous nous sommes penchés en détail sur les données structurées générées périodiquement que les bailleurs sociaux possèdent. Nous avons focalisé notre travail d'aide à la création de métadonnées pour ce type de données, en descendant au niveau « attribut » (et non en restant au niveau « fichier / table »), pour connaître de manière plus fine quelles sont les différences entre les occurrences de données. En ayant connaissance de ces différences, l'utilisateur peut ensuite savoir aisément quelles transformations sont à effectuer pour que plusieurs occurrences de données de la même référence correspondent à un même schéma. À nouveau, par expérience chez BIAL-X, nous avons pu constater que le plus complexe pour transformer des données est de savoir exactement

les étapes à réaliser, mais pas de les effectuer de manière concrète. Avec une description suffisamment détaillée des attributs d'entités de données structurées, l'utilisateur peut alors facilement déterminer les étapes à mettre en place pour transformer les données afin qu'elles correspondent à un même schéma. Tout ceci permet de fournir aux utilisateurs des données en « libre service » et leur laisse la possibilité de les utiliser pour leurs besoins d'analyse.

De plus, en qualifiant suffisamment bien les données à leur ingestion, il est possible de savoir quelles données contiennent quoi, c'est-à-dire quels attributs sont présents dans les données. Ensuite, l'utilisateur peut récupérer toutes les entités de données qui contiennent un attribut portant une information spécifique, nécessaire pour un besoin d'analyse donné. Il peut aussi connaître s'il y a des différences de codage entre les attributs provenant d'entités de données différentes : par exemple, les modalités d'un attribut « sexe » peuvent être `H` et `F` dans une entité de données, mais `1` et `2` dans une autre. Avec ceci, il n'est plus nécessaire de savoir exactement combien d'entités de données sont concernées : l'utilisateur sait simplement quelles opérations appliquer pour que toutes les données se conforment au besoin ponctuel. Ainsi, nous rationalisons le travail qui doit être effectué pour modifier les données, en particulier dans la phase d'étude. Précisons que les métadonnées ne sont là que pour aider l'utilisateur si le besoin d'analyse qu'il cherche à traiter requiert de transformer des données du lac ; les données restent néanmoins toujours disponibles dans leur état d'origine, que l'utilisateur les transforme ou non.

Dans la section précédente, nous avons constaté que HOUDAL permet aux bailleurs sociaux d'adresser de nouveaux usages, en laissant la possibilité de créer de nouveaux flux d'exploitation de données sans venir perturber l'existant. QSTR va plus loin en permettant de réellement pérenniser ces nouveaux usages, i.e. de les intégrer plus facilement aux autres exploitations de données existantes. En effet, vu que les évolutions de schéma au fil des occurrences de données sont amorties grâce aux métadonnées générées à l'aide de QSTR, le traitement métier (qui transforme les données pour un besoin d'analyse défini) n'évolue que très peu. À titre d'exemple, si un attribut change de codage dans une nouvelle occurrence de données, l'utilisateur est alors prévenu, ce qui permet de modifier immédiatement le traitement métier et ce de manière assez simple et rapide.

Le second avantage majeur apporté par l'assistant de création de métadonnées est qu'en qualifiant les données externes d'une manière détaillée (au niveau « attribut »), il est possible de les mettre à disposition dans des offres commerciales de l'entreprise. Une description détaillée des données permet de mieux comprendre leur contenu, ce qui rend possible la mise à disposition de ces données en libre service aux utilisateurs, pour les laisser les exploiter pour un besoin donné. Notons qu'il est en réalité déjà possible d'y inclure des données externes, en dehors de HOUDAL et de son assistant QSTR. Toutefois, sans ceux-ci, l'étude puis la transformation de ces données sont généralement effectuées une seule fois, et elles requièrent un fort investissement de la part de l'entreprise et/ou du bailleur social, tout en étant très difficilement réutilisable. Nos contributions permettent donc à la fois d'adresser plus facilement un nouveau besoin donné, mais aussi de rendre pérenne les résultats d'analyse obtenus, s'ils sont suffisamment satisfaisants.

Évidemment, la qualification de données provenant de sources externes nécessite toujours un investissement, mais QSTR facilite le travail pour qualifier les données d'une manière complète et réutilisable. De plus, les transformations ne sont effectuées que si l'utilisateur estime que c'est nécessaire : la création de métadonnées n'altère pas les données. Cette approche constitue un véritable changement dans la manière de stocker les données puis de les utiliser. Cela n'empêche pas de transformer les données après coup pour un besoin ponctuel, mais cette transformation est facilitée grâce aux métadonnées, puisque la tâche la plus complexe est l'étude des données pour savoir quelle(s) transformation(s) sont à effectuer pour répondre à un besoin d'analyse donné.

Pour conclure, les travaux menés au cours de cette thèse apportent des contributions à la thématique des lacs de données, et répondent à des besoins métier. Mais ces contributions s'accompagnent aussi de nouvelles questions.

6.2 Perspectives de recherche

Nous identifions un total de cinq perspectives de recherche qui découlent de nos contributions, et elles pourraient faire l'objet de travaux futurs. Bien qu'elles ne soient pas opposées, nous distinguons ces perspectives en deux catégories : les perspectives industrielles d'une part, qui s'inscrivent directement dans le contexte de l'entreprise, tout en soulevant des aspects scientifiques (section 6.2.1) ; et les perspectives académiques d'autre part, qui s'ancrent d'emblée dans une dimension plus recherche (section 6.2.2).

6.2.1 Perspectives industrielles

Les trois perspectives que nous listons sont soit en cours de réalisation chez BIAL-X, soit prévues dans le cadre de nouveaux projets.

6.2.1.1 Industrialisation et modularisation de HOUDAL

Nos propositions académiques, avec d'une part le métamodèle de métadonnées goldMEDAL et d'autre part l'architecture de lac de données, ont été mises en œuvre sous la forme d'une preuve de concept pour donner naissance à HOUDAL. Cette proposition illustre tout l'intérêt d'un lac de données pour gérer des problèmes qui surviennent lors de l'analyse de données variées avec des outils traditionnels. Mais cette preuve de concept n'est pas encore passée à l'étape supérieure pour l'entreprise, à savoir une solution industrialisée et commercialisable, bien que plusieurs bailleurs sociaux avec qui BIAL-X travaille aient fait part de leur intérêt pour HOUDAL.

Ainsi, l'industrialisation de HOUDAL est un enjeu majeur pour BIAL-X, qui souhaite pouvoir déployer le lac de données chez un ou plusieurs bailleur(s). Cependant, le déploiement manuel de toute l'infrastructure de HOUDAL est un travail fastidieux et chronophage, et c'est pourquoi nous nous intéressons au mouvement *DevOps* et au principe de l'*Infrastructure as Code* (IaC) dans l'optique d'automatiser autant que possible le déploiement de notre solution. Pour ce faire, nous aimerions proposer un lac de données

modulaire, comme une boite à outils où les bailleurs piochent dans les différents modules disponibles. Alors, il serait possible de sélectionner les modules dont le bailleur a besoin, puis que le déploiement s'effectue automatiquement.

Cette problématique a récemment motivé le démarrage d'un projet d'ampleur au sein de la DIA avec le lancement d'une nouvelle entité nommée BIAL-S aux côtés de BIAL-X et BIAL-R. Un des objectifs de ce projet est de créer une nouvelle offre commerciale destinée aux bailleurs sociaux (dans un premier temps, qui pourrait être généralisée par la suite) proposant de déployer une plateforme complète de gestion de données, de l'entreposage jusqu'à l'extraction de connaissances. Cette plateforme repose grandement sur le principe de l'IaC qui fournit un déploiement automatisé de toute l'infrastructure. La plateforme vise à proposer divers services, et parmi eux se trouve la création d'un système de métadonnées basé sur celui qui a été implémenté dans HOUDAL.

6.2.1.2 Enrichissement et généralisation de QSTR

La création assistée ou semi-automatique de métadonnées au moment de l'insertion de nouvelles entités de données dans le lac de données est une problématique à laquelle nous avons apporté une contribution avec QSTR. C'est un assistant à la création de métadonnées pour les données structurées générées périodiquement, qui compare les occurrences de données déjà enregistrées dans le lac aux nouvelles données que l'utilisateur souhaite enregistrer.

Toutefois, QSTR se limite aux données structurées générées périodiquement, et nous avons pour objectif de dépasser cette limite et de le généraliser pour assister l'utilisateur à qualifier tous types de données. Bien entendu, les méthodes employées dans QSTR pour extraire des informations de données structurées ne peuvent pas être reproduites à l'identique sur des données semi-structurées ou non structurées. La littérature propose diverses approches pour l'extraction automatique de métadonnées spécifiques selon le type des données : semi-structurées, textuelles, audio, images, etc. Nous souhaitons donc combiner ces méthodes existantes au sein de QSTR pour le généraliser, et pourquoi pas rejoindre la perspective de modularisation du lac de données en proposant divers modules au sein de l'assistant. Au déploiement du lac de données, il serait alors possible de ne prendre que les sous-modules de QSTR qui sont nécessaires pour répondre à un besoin métier donné, sans avoir à déployer l'entièreté de l'assistant.

Cette perspective vise aussi à mettre l'accent sur le composant d'ingestion de notre proposition d'architecture de lac de données, et du principe *describe-on-write*. Nous pensons qu'un assistant comme QSTR est la condition nécessaire à la pérennité de HOUDAL, car alimenter correctement le métamodèle goldMEDAL manuellement est une tâche fastidieuse sur le long terme. L'enjeu est de faire en sorte que les utilisateurs du lac de données adhèrent bien à la philosophie de celui-ci : pour ce faire, nous cherchons à leur faciliter le plus possible le travail de création des métadonnées pour que cette étape cruciale soit correctement menée à bien.

6.2.1.3 Interrogation des données à partir de requêtes sur les métadonnées

La dernière perspective industrielle que nous relevons concerne l'interrogation des données du lac sur la base de requêtes formulées sur les métadonnées. Ces dernières décrivent les données, et sont à un niveau d'abstraction plus élevé. Par exemple, pour des données structurées, il est souvent plus compréhensible de lire la désignation fonctionnelle d'un attribut plutôt que son intitulé, qui suit parfois des règles de nommage assez techniques. Si cette information est présente dans les métadonnées, il est pertinent de la fournir à l'utilisateur pour faciliter sa compréhension des données présentes dans le lac.

L'enjeu est ici de proposer un outil d'interrogation qui s'intègre au composant d'exploitation de notre architecture de lac de données et puisse devenir à terme un point d'exploitation majeur du lac. Nous souhaitons que l'utilisateur puisse formuler une requête en se basant exclusivement sur les métadonnées du lac, c'est-à-dire des informations à un niveau d'abstraction plus élevé que les données elles-mêmes et généralement plus compréhensibles. Le système devrait alors être capable d'interroger les métadonnées pour déterminer quelles données répondent à la requête de l'utilisateur, pour lui fournir *in fine* un résultat potentiellement constitué de données hétérogènes. Dans le cadre de l'habitat social, un exemple de requête serait d'obtenir toutes les données qui sont liées à un logement donné, spécifié par l'utilisateur. Le système serait alors à même de renvoyer des données hétérogènes : par exemple, des données structurées qui contiennent les propriétés intrinsèques dudit logement, mais aussi des photos de celui-ci, ou encore des documents textuels qui pourraient être des factures de travaux effectués dans le logement.

Évidemment, cet objectif soulève plusieurs problèmes, à commencer par l'intégration de cet outil d'interrogation à l'interface de gestion du lac de données. De plus, la recherche d'information n'est pas la seule façon d'exploiter le lac, et il reste à étudier comment assembler cette fonctionnalité à celle d'exploration des métadonnées, où l'utilisateur navigue parmi les informations disponibles.

6.2.2 Perspectives académiques

En plus des perspectives industrielles, nous présentons deux perspectives académiques plutôt orientées recherche, qui sont un peu plus éloignées des enjeux de l'entreprise à l'heure actuelle.

6.2.2.1 Rendre le lac de données plus inclusif

Une première perspective académique porte sur l'ouverture des lacs de données à des personnes qui ne sont pas des spécialistes des données ou de l'informatique. C'est un point que nous avions déjà mentionné dans notre notre définition du concept de lac de données, où contrairement à d'autres propositions de la littérature, nous stipulons qu'un lac est dédié *principalement* à des spécialistes des données, sans pour autant fermer l'accès à des utilisateurs ayant des profils différents. L'idée de cette perspective de recherche est d'aller encore plus loin dans cette idée en retirant toute notion de destinataire dans la définition de lac de données. Ceci passe par le fait d'ouvrir complètement l'accès au lac

de données à des utilisateurs qui ne vont pas interagir avec le lac de données de la même manière que des *data scientists* ou des *data engineers*, mais plutôt comme des experts métiers ou encore des dirigeants.

Cette perspective soulève plusieurs questions. Dans un premier temps, en modifiant la définition de lac de données pour l'ouvrir à tous types d'utilisateurs, quels sont les impacts sur la conception, la modélisation et l'implémentation d'un lac ? Nous pouvons imaginer que cela passe par la création d'une couche logicielle, connectée au système de gestion des métadonnées, qui permet à ces nouveaux utilisateurs de transformer et d'analyser leurs propres données en toute autonomie. Toutefois, une telle couche logicielle ne doit pas devenir une boîte noire de plus : il faut veiller à accompagner les utilisateurs dans leur appropriation des outils d'analyse, non seulement par la formation, mais aussi en imbriquant les méthodologies de recherche issues de l'informatique avec les pratiques d'entreprise en étroite collaboration avec les partenaires.

6.2.2.2 Qualité des métadonnées

Les travaux de ce manuscrit témoignent de l'importance capitale des métadonnées pour le bon fonctionnement d'un lac de données. Il devient alors vital de les considérer avec la même importance que les spécialistes des données considèrent les données qu'ils manipulent pour leurs besoins d'analyse. L'étude de la qualité des données est une thématique de recherche largement abordée dans la littérature scientifique [Batini et al., 2006, Zaveri et al., 2016]. Nous pensons qu'il serait judicieux de chercher à appliquer ces principes aux métadonnées, dans l'optique de pouvoir étudier la qualité des métadonnées d'un lac de données. Nous pourrions, par exemple, confronter les métadonnées du lac aux dimensions qualité couramment utilisées, comme l'exactitude, la complétude, la redondance, la consistance, la lisibilité ou la confiance.

Compte tenu de toutes ces perspectives de recherche et les questions restées en suspens, une nouvelle thèse CIFRE démarre en collaboration entre BIAL-X et le laboratoire ERIC.

Bibliographie

[Allan et al., 2000] Allan, J., Lavrenko, V., Malin, D., and Swan, R. (2000). Detections, bounds, and timelines : Umass and tdt-3. In *Topic Detection and Tracking Workshop (TDT-3), Vienna, VA, USA*, pages 167–174.

[Alrehamy and Walker, 2015] Alrehamy, H. and Walker, C. (2015). Personal Data Lake With Data Gravity Pull. In *IEEE 5th International Conference on Big Data and Cloud Computing (BDCloud 2015), Dalian, China*, volume 88 of *IEEE Computer Society Washington*, pages 160–167.

[Ansari et al., 2018] Ansari, J. W., Karim, N., Decker, S., Cochez, M., and Beyan, O. (2018). Extending Data Lake Metadata Management by Semantic Profiling. In *2018 Extended Semantic Web Conference (ESWC 2018), Heraklion, Crete, Greece*, ESWC, pages 1–15.

[Armbrust et al., 2020] Armbrust, M., Das, T., Sun, L., Yavuz, B., Zhu, S., Murthy, M., Torres, J., van Hovell, H., Ionescu, A., Łuszczak, A., et al. (2020). Delta lake : high-performance acid table storage over cloud object stores. *Proceedings of the Very Large Data Base Endowment (VLDB 2020)*, 13(12) :3411–3424.

[Armbrust et al., 2021] Armbrust, M., Ghodsi, A., Xin, R., and Zaharia, M. (2021). Lakehouse : A new generation of open platforms that unify data warehousing and advanced analytics. In *11th Conference on Innovative Data Systems Research (CIDR 2021)*. www.cidrdb.org.

[Assunção et al., 2015] Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A., and Buyya, R. (2015). Big data computing and clouds : Trends and future directions. *Journal of Parallel and Distributed Computing*, 79 :3–15.

[Baars and Ereth, 2016] Baars, H. and Ereth, J. (2016). From data warehouses to analytical atoms - the internet of things as a centrifugal force in business intelligence and analytics. In *24th European Conference on Information Systems (ECIS 2016), Istanbul, Turkey*.

[Batini et al., 2006] Batini, C., Scannapieco, M., et al. (2006). *Data Quality : Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer.

[Beheshti et al., 2017] Beheshti, A., Benatallah, B., Nouri, R., Chhieng, V. M., Xiong, H., and Zhao, X. (2017). CoreDB : a Data Lake Service. In *2017 ACM on Conference on Information and Knowledge Management (CIKM 2017), Singapore, Singapore*, ACM, pages 2451–2454.

[Beheshti et al., 2018] Beheshti, A., Benatallah, B., Nouri, R., and Tabebordbar, A. (2018). CoreKG : A Knowledge Lake Service. *Proceedings of the Very Large Data Base Endowment (VLDB 2018)*, 11(12) :1942–1945.

[Bicevska and Oditis, 2017] Bicevska, Z. and Oditis, I. (2017). Towards nosql-based data warehouse solutions. *Procedia Computer Science*, 104 :104–111.

[Bogatu et al., 2020] Bogatu, A., Fernandes, A. A., Paton, N. W., and Konstantinou, N. (2020). Dataset discovery in data lakes. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 709–720. IEEE.

[Bouaziz et al., 2019] Bouaziz, S., Nabli, A., and Gargouri, F. (2019). Design a data warehouse schema from document-oriented database. *Procedia Computer Science*, 159 :221–230.

[Boussahoua et al., 2017] Boussahoua, M., Boussaid, O., and Bentayeb, F. (2017). Logical schema for data warehouse on column-oriented nosql databases. In *International Conference on Database and Expert Systems Applications*, pages 247–256. Springer.

[Chen et al., 2012] Chen, H., Chiang, R. H., and Storey, V. C. (2012). Business intelligence and analytics : from big data to big impact. *MIS quarterly*, pages 1165–1188.

[Chen et al., 2014] Chen, M., Mao, S., and Liu, Y. (2014). Big data : A survey. *Mobile networks and applications*, 19(2) :171–209.

[Davoudian et al., 2018] Davoudian, A., Chen, L., and Liu, M. (2018). A survey on nosql stores. *ACM Computing Surveys (CSUR)*, 51(2) :1–43.

[Dehghani, 2019] Dehghani, Z. (2019). How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh . https ://martinfowler.com/articles/data-monolith-to-mesh.html.

[Dehghani, 2020] Dehghani, Z. (2020). Data Mesh Principles and Logical Architecture. https ://martinfowler.com/articles/data-mesh-principles.html.

[Diamantini et al., 2018] Diamantini, C., Giudice, P. L., Musarella, L., Potena, D., Storti, E., and Ursino, D. (2018). A New Metadata Model to Uniformly Handle Heterogeneous Data Lake Sources. In *European Conference on Advances in Databases and Information Systems (ADBIS 2018), Budapest, Hungary*, pages 165–177.

[Ding et al., 2004] Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y., Reddivari, P., Doshi, V., and Sachs, J. (2004). Swoogle : a search and metadata engine for the semantic web. In *Proceedings of the 2004 ACM International Conference on Information and Knowledge Management (CIKM 2004), Washington DC, USA*, pages 652–659.

[Dixon, 2010] Dixon, J. (2010). Pentaho, Hadoop, and Data Lakes. https ://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/.

[Dutta and Dutta, 2019] Dutta, P. and Dutta, P. (2019). Comparative study of cloud services offered by amazon, microsoft & google. *International Journal of Trend in Scientific Research and Development (IJTSRD)*, 3(3) :981–985.

[Eichler et al., 2020] Eichler, R., Giebler, C., Gröger, C., Schwarz, H., and Mitschang, B. (2020). HANDLE-A Generic Metadata Model for Data Lakes. In *International Conference on Big Data Analytics and Knowledge Discovery (DaWak 2020), Bratislava, Slovakia*, pages 73–88.

[Eichler et al., 2021] Eichler, R., Giebler, C., Gröger, C., Schwarz, H., and Mitschang, B. (2021). Modeling metadata in data lakes—a generic model. *Data & Knowledge Engineering*, page 101931.

[Fang, 2015] Fang, H. (2015). Managing Data Lakes in Big Data Era : What's a data lake and why has it became popular in data management ecosystem. In *5th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems (CYBER 2015), Shenyang, China*, IEEE, pages 820–824.

[Farid et al., 2016] Farid, M., Roatis, A., Ilyas, I. F., Hoffmann, H.-F., and Chu, X. (2016). CLAMS : Bringing Quality to Data Lakes. In *2016 International Conference on Management of Data (SIGMOD 2016), San Francisco, CA, USA*, ACM, pages 2089–2092.

[Farrugia et al., 2016] Farrugia, A., Claxton, R., and Thompson, S. (2016). Towards Social Network Analytics for Understanding and Managing Enterprise Data Lakes. In *Advances in Social Networks Analysis and Mining (ASONAM 2016), San Francisco, CA, USA*, IEEE, pages 1213–1220.

[Fauduet and Peyrard, 2010] Fauduet, L. and Peyrard, S. (2010). A Data-First Preservation Strategy : Data Management In SPAR. In *7th International Conference on Preservation of Digital Objects (iPRES 2010), Vienna, Autria*, pages 1–8.

[Fernandez et al., 2018] Fernandez, R. C., Abedjan, Z., Koko, F., Yuan, G., Madden, S., and Stonebraker, M. (2018). Aurum : A data discovery system. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1001–1012. IEEE.

[Gandomi and Haider, 2015] Gandomi, A. and Haider, M. (2015). Beyond the hype : Big data concepts, methods, and analytics. *International Journal of Information Management*, 35(2) :137–144.

[Giebler et al., 2019] Giebler, C., Gröger, C., Hoos, E., Schwarz, H., and Mitschang, B. (2019). Leveraging the Data Lake - Current State and Challenges. In *International Conference on Big Data Analytics and Knowledge Discovery (DaWaK 2019), Linz, Austria*.

[Giudice et al., 2018] Giudice, P. L., Musarella, L., Sofo, G., and Ursino, D. (2018). An approach to extracting complex knowledge patterns among concepts belonging to structured, semi-structured and unstructured sources in a data lake. *Information Sciences*, 478 :606–626.

[Gröger, 2018] Gröger, C. (2018). Building an industry 4.0 analytics platform. *Datenbank-Spektrum*, 18(1) :5–14.

[Gupta et al., 2017] Gupta, A., Tyagi, S., Panwar, N., Sachdeva, S., and Saxena, U. (2017). Nosql databases : Critical analysis and comparison. In *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, pages 293–299. IEEE.

[Hai et al., 2016] Hai, R., Geisler, S., and Quix, C. (2016). Constance : An Intelligent Data Lake System. In *International Conference on Management of Data (SIGMOD 2016), San Francisco, CA, USA*, ACM Digital Library, pages 2097–2100.

[Halevy et al., 2016] Halevy, A., Korn, F., Noy, N. F., Olston, C., Polyzotis, N., Roy, S., and Whang, S. E. (2016). Managing Google's data lake : an overview of the GOODS system. In *2016 International Conference on Management of Data (SIGMOD 2016), San Francisco, CA, USA*, ACM, pages 795–806.

[Haste, 2017] Haste, J.-L. (2017). From the data lake to the agile data warehouse : decision-making in the big data era . https ://en.blog.businessdecision.com/bigdata-en/2017/02/data-lake-data-warehouse-big-data-era/.

[Heintz and Lee, 2019] Heintz, B. and Lee, D. (2019). Productionizing Machine Learning with Delta Lake. https ://databricks.com/fr/blog/2019/08/14/productionizing-machine-learning-with-delta-lake.html.

[Hellerstein et al., 2017] Hellerstein, J. M., Sreekanti, V., Gonzalez, J. E., Dalton, J., Dey, A., Nag, S., Ramachandran, K., Arora, S., Bhattacharyya, A., Das, S., Donsky, M., Fierro, G., She, C., Steinbach, C., Subramanian, V., and Sun, E. (2017). Ground : A Data Context Service. In *Biennial Conference on Innovative Data Systems Research (CIDR 2017), Chaminade, CA, USA*.

[Inmon, 2016] Inmon, B. (2016). *Data Lake Architecture : Designing the Data Lake and avoiding the garbage dump*. Technics Publications.

[Inmon, 1996] Inmon, W. H. (1996). *Building the Data Warehouse*. John Wiley & Sons.

[Jaccard, 1912] Jaccard, P. (1912). The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2) :37–50.

[John and Misra, 2017] John, T. and Misra, P. (2017). *Data Lake for Enterprises : Lambda Architecture for building enterprise data systems*. Packt Publishing.

[Kamal et al., 2020] Kamal, M. A., Raza, H. W., Alam, M. M., and Su'ud, M. M. (2020). Highlight the features of aws, gcp and microsoft azure that have an impact when choosing a cloud service provider. *International Journal of Recent Technology and Engineering (IJRTE)*.

[Khine and Wang, 2017] Khine, P. P. and Wang, Z. S. (2017). Data Lake : A New Ideology in Big Data Era. In *International Conference on Wireless Communication and Sensor Network (WCSN 2017), Wuhan, China*, volume 17 of *ITM Web of Conferences*, pages 1–6.

[Kimball, 2008] Kimball, R. (2008). Slowly changing dimensions. *Information Management*, 18(9) :29.

[Klettke et al., 2017] Klettke, M., Awolin, H., Stürl, U., Müller, D., and Scherzinger, S. (2017). Uncovering the Evolution History of Data Lakes. In *2017 IEEE International Conference on Big Data (BIGDATA 2017), Boston, MA, USA*, pages 2462–2471.

[Kondrak, 2005] Kondrak, G. (2005). N-gram similarity and distance. In *International symposium on string processing and information retrieval*, pages 115–126. Springer.

[Krishnan, 2013] Krishnan, K. (2013). *Data warehousing in the age of big data*. Morgan Kaufmann.

[Kruskal and Wallis, 1952] Kruskal, W. H. and Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260) :583–621.

[LaPlante and Sharma, 2016] LaPlante, A. and Sharma, B. (2016). Architecting data lakes data management architectures for advanced business use cases.

[Laskowski, 2016] Laskowski, N. (2016). Data lake governance : A big data do or die. https ://searchcio.techtarget.com/feature/Data-lake-governance-A-big-data-do-or-die.

[Le and Mikolov, 2014] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR.

[Levenshtein et al., 1966] Levenshtein, V. I. et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

[Liu et al., 2021] Liu, P., Loudcher, S., Darmont, J., and Noûs, C. (2021). ArchaeoDAL : A Data Lake for Archaeological Data Management and Analytics. In *25th International Database Engineering and Applications Symposium (IDEAS 2021), Montreal, Canada*, pages 252–262.

[Maccioni and Torlone, 2017] Maccioni, A. and Torlone, R. (2017). Crossing the finish line faster when paddling the data lake with KAYAK. *Proceedings of the Very Large Data Base Endowment (VLDB 2017)*, 10(12) :1853–1856.

[Maccioni and Torlone, 2018] Maccioni, A. and Torlone, R. (2018). KAYAK : A Framework for Just-in-Time Data Preparation in a Data Lake. In *International Conference on Advanced Information Systems Engineering (CAiSE 2018), Tallin, Estonia*, pages 474–489.

[Machado et al., 2021] Machado, I., Costa, C., and Santos, M. Y. (2021). Data-driven information systems : The data mesh paradigm shift. In *Information Systems Development : Crossing Boundaries between Development and Operations (DevOps) in Information Systems (ISD2021 Proceedings), Valencia, Spain.*

[Madera and Laurent, 2016] Madera, C. and Laurent, A. (2016). The next information architecture evolution : the data lake wave. In *International Conference on Management of Digital EcoSystems (MEDES 2016), Biarritz, France*, pages 174–180.

[Massey Jr, 1951] Massey Jr, F. J. (1951). The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253) :68–78.

[Mathis, 2017] Mathis, C. (2017). Data Lakes. *Datenbank-Spektrum*, 17(3) :289–293.

[Maurya et al., 2021] Maurya, S., Mufti, T., Kumar, D., Mittal, P., and Gupta, R. (2021). A study on cloud computing : A review. In *Proceedings of the 2nd International Conference on ICT for Digital, Smart, and Sustainable Development (ICIDSSD 2020).*

[Mehmood et al., 2019] Mehmood, H., Gilman, E., Cortes, M., Kostakos, P., Byrne, A., Valta, K., Tekes, S., and Riekki, J. (2019). Implementing big data lake for heterogeneous data sources. In *International Conference on Data Engineering Workshops (ICDEW 2019), Macau SAR, China*, IEEE, pages 37–44.

[Miloslavskaya and Tolstoy, 2016] Miloslavskaya, N. and Tolstoy, A. (2016). Big Data, Fast Data and Data Lake Concepts. In *International Conference on Biologically Inspired Cognitive Architectures (BICA 2016), NY, USA*, volume 88 of *Procedia Computer Science*, pages 1–6.

[Mortenson et al., 2015] Mortenson, M. J., Doherty, N. F., and Robinson, S. (2015). Operational research from taylorism to terabytes : A research agenda for the analytics age. *European Journal of Operational Research*, 241(3) :583–595.

[O'Leary, 2014] O'Leary, D. E. (2014). Embedding AI and Crowdsourcing in the Big Data Lake. *IEEE Intelligent Systems*, 29(5) :70–73.

[Oram, 2015] Oram, A. (2015). *Managing the Data Lake : Moving to Big Data Analysis.* O'Reilly Media.

[Pathirana, 2015] Pathirana, N. (2015). Modeling Industrial and Cultural Heritage Data. Master's thesis, Université Lumière Lyon 2, France.

[Pearson, 1900] Pearson, K. (1900). X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302) :157–175.

[Quix and Hai, 2018] Quix, C. and Hai, R. (2018). Data Lake. *Encyclopedia of Big Data Technologies*, pages 1–8.

[Quix et al., 2016] Quix, C., Hai, R., and Vatov, I. (2016). Metadata Extraction and Management in Data Lakes With GEMMS. *Complex Systems Informatics and Modeling Quarterly*, 9 :67–83.

[Ravat and Zhao, 2019a] Ravat, F. and Zhao, Y. (2019a). Data lakes : Trends and perspectives. In *International Conference on Database and Expert Systems Applications (DEXA 2019), Linz, Austria*, pages 304–313. Springer.

[Ravat and Zhao, 2019b] Ravat, F. and Zhao, Y. (2019b). Metadata management for data lakes. In *European Conference on Advances in Databases and Information Systems (ADBIS 2019), Bled, Slovenia*, pages 37–44. Springer.

[Riley, 2017] Riley, J. (2017). Understanding metadata : What is metadata, and what is it for ? *National Information Standards Organization (NISO).*

[Sawadogo and Darmont, 2021] Sawadogo, P. and Darmont, J. (2021). On data lake architectures and metadata management. *Journal of Intelligent Information Systems*, 56(1) :97–120.

[Sawadogo et al., 2019a] Sawadogo, P., Kibata, T., and Darmont, J. (2019a). Metadata management for textual documents in data lakes. *21st International Conference on Enterprise Information Systems (ICEIS 2019), Heraklion, Crete, Greece*, pages 72–83.

[Sawadogo et al., 2021] Sawadogo, P. N., Darmont, J., and Noûs, C. (2021). Joint Management and Analysis of Textual Documents and Tabular Data within the AUDAL Data Lake. In *25th European Conference on Advances in Databases and Information Systems (ADBIS 2021), Tartu, Estonia*, pages 88–101. Springer.

[Sawadogo et al., 2019b] Sawadogo, P. N., Scholly, E., Favre, C., Ferey, E., Loudcher, S., and Darmont, J. (2019b). Metadata systems for data lakes : models and features. In *International Workshop on BI and Big Data Applications (BBIGAP@ADBIS 2019), Bled, Slovenia*, pages 440–451. Springer.

[Schneider et al., 2019] Schneider, S., Baevski, A., Collobert, R., and Auli, M. (2019). wav2vec : Unsupervised pre-training for speech recognition. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 3465–3469. ISCA.

[Scholly et al., 2021a] Scholly, É., Favre, C., Ferey, É., and Loudcher, S. (2021a). HOUDAL : A data lake implemented for public housing. In *Proceedings of the 23rd International Conference on Enterprise Information Systems, ICEIS 2021, Online Streaming*, volume 1, pages 39–50. SCITEPRESS.

[Scholly et al., 2021b] Scholly, E., Sawadogo, P., Liu, P., Espinosa-Oviedo, J. A., Favre, C., Loudcher, S., Darmont, J., and Noûs, C. (2021b). Coining goldmedal : A new contribution to data lake generic metadata modeling. In *23rd International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP@ EDBT 2021)*.

[Scholly et al., 2019] Scholly, E., Sawadogo, P. N., Favre, C., Ferey, E., Loudcher, S., and Darmont, J. (2019). Systèmes de métadonnées dans les lacs de données : modélisation et fonctionnalités. In *15e journées EDA Business Intelligence & Big Data (EDA 2019)*, pages 77–92.

[Singh et al., 2016] Singh, K., Paneri, K., Pandey, A., Gupta, G., Sharma, G., Agarwal, P., and Shroff, G. (2016). Visual Bayesian Fusion to Navigate a Data Lake. In *19th International Conference on Information Fusion (FUSION 2016), Heidelberg, Germany*, IEEE, pages 987–994.

[Sirosh, 2016] Sirosh, J. (2016). The Intelligent Data Lake. https ://azure.microsoft.com/fr-fr/blog/the-intelligent-data-lake/.

[Sivarajah et al., 2017] Sivarajah, U., Kamal, M. M., Irani, Z., and Weerakkody, V. (2017). Critical analysis of big data challenges and analytical methods. *Journal of Business Research*, 70 :263–286.

[Spearman, 1961] Spearman, C. (1961). The proof and measurement of association between two things. *Studies in individual differences : The search for intelligence*, pages 45–58.

[Stefanowski et al., 2017] Stefanowski, J., Krawiec, K., and Wrembel, R. (2017). Exploring Complex and Big Data. *International Journal of Applied Mathematics and Computer Science*, 27(4) :669–679.

[Subramaniam et al., 2021] Subramaniam, P., Ma, Y., Li, C., Mohanty, I., and Fernandez, R. C. (2021). Comprehensive and comprehensible data catalogs : The what, who, where, when, why, and how of metadata management. *CoRR*, abs/2103.07532.

[Suriarachchi and Plale, 2016] Suriarachchi, I. and Plale, B. (2016). Crossing Analytics Systems : A Case for Integrated Provenance in Data Lakes. In *International Conference on e-Science (e-Science 2016), Baltimore, MD, USA*, IEEE, pages 349–354.

[Szymkiewicz, 1934] Szymkiewicz, D. (1934). Une contribution statistique à la géographie floristique. *Acta Societatis Botanicorum Poloniae*, 11(3) :249–265.

[Tate, 1954] Tate, R. F. (1954). Correlation between a discrete and a continuous variable. point-biserial correlation. *The Annals of mathematical statistics*, 25(3) :603–607.

[Terrizzano et al., 2015] Terrizzano, I., Schwarz, P., Roth, M., and Colino, J. E. (2015). Data Wrangling : The Challenging Journey from the Wild to the Lake. In *7th Biennial Conference on Innovative Data Systems Research (CIDR 2015), Asilomar, CA, USA*, pages 1–9.

[Valnaos, 2019] Valnaos (2019). AWS, Azure, Google : Quel cloud choisir ? https ://www.valnaos.com/aws-azure-google-quel-cloud-choisir/.

[Watson and Wixom, 2007] Watson, H. J. and Wixom, B. H. (2007). The current state of business intelligence. *Computer*, 40(9) :96–99.

[Wirth and Hipp, 2000] Wirth, R. and Hipp, J. (2000). Crisp-dm : Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, volume 1, pages 29–40. Manchester.

[Wrembel and Bębel, 2007] Wrembel, R. and Bębel, B. (2007). Metadata management in a multiversion data warehouse. In *Journal on data semantics VIII*, pages 118–157. Springer.

[Zaveri et al., 2016] Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., and Auer, S. (2016). Quality assessment for linked data : A survey. *Semantic Web*, 7(1) :63–93.

[Zhao et al., 2021] Zhao, Y., Megdiche, I., and Ravat, F. (2021). Data lake ingestion management. *CoRR*, abs/2107.02885.

Table des figures

Liste des tableaux

Table des matières