Des lacs de données à l'analyse assistée de documents textuels et tabulaires

Pegdwendé Sawadogo

N° d'ordre NNT : 2021LYSE2088

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 20 octobre 2021, par :

Pegdwendé SAWADOGO

Des lacs de données à l'analyse assistée de documents textuels et tabulaires.

Devant le jury composé de :

Sabine LOUDCHER, Professeure des universités, Université Lumière Lyon 2, Présidente

Franck RAVAT, Professeur des universités, Université Toulouse 1, Rapporteur

Anne LAURENT, Professeure d'université, Université de Montpellier, Rapporteure

Martine SÉVILLE, Professeure des universités, Université Lumière Lyon 2, Examinatrice

Esteban ZIMANYI, Professeur, Université libre de Bruxelles, Examinateur

Ioana MANOLESCU, Directrice de recherche, INRIA Saclay Île-de-France, Examinatrice

Jérôme DARMONT, Professeur des universités, Université Lumière Lyon 2, Directeur de thèse

Contrat de diffusion

Université de Lyon

Ecole Doctorale Informatique & Mathématiques, ED 512

THÈSE

Pour obtenir le grade de

Docteur en Informatique

Réalisée au sein de l'Unité de Recherche ERIC

Des lacs de données à l'analyse assistée de documents textuels et tabulaires

Par

Pegdwendé Nicolas SAWADOGO

Sous la direction de

Jérôme DARMONT

Présentée et soutenue publiquement le 20 Octobre 2021

Devant le jury composé de:

Anne LAURENT	Rapportrice	Professeure des universités, Université de Montpellier
Franck RAVAT	Rapporteur	Professeur des universités, Université Toulouse I Capitole
Sabine LOUDCHER	Examinatrice	Professeure des universités, Université Lumière Lyon 2
Ioana MANOLESCU	Examinatrice	Directrice de recherche, INRIA - Saclay
Martine SÉVILLE	Examinatrice	Professeure des universités, Université Lumière Lyon 2
Esteban ZIMÁNYI	Examinateur	Professeur des universités, Université Libre de Bruxelles
Jérôme DARMONT	Directeur de thèse	Professeur des universités, Université Lumière Lyon 2

Abstract

Over the past decade, the concept of a data lake has emerged as an alternative to data warehouses for big data storage and analysis. Data lakes follow a schema-on-read approach to provide flexible and scalable decision-support systems. A key issue in data lakes is the need of an effective metadata system. In the absence of a fixed data schema, metadata are indeed essential to support analyses and thus prevent the lake from turning into an inoperable data swamp.

Though the literature seems unanimous on the importance of metadata systems, there are still many questions and uncertainties about their implementation methodology. Several approaches have been proposed to organize metadata in data lakes, but most of them do not support industrialized analyses as in data warehouses. In addition, a significant part of the literature limits access to the data lake to data scientists, thus excluding business users. Moreover, a wide majority of existing metadata management approaches in data lakes only concern structured and semi-structured data. Designing a metadata system that supports both industrialized analyses and unstructured data is therefore still an open research issue.

In this context, we propose through this thesis a set of contributions to the literature on the design and implementation of data lakes. Our contributions are divided into three parts. The first part targets the disambiguation of the data lake concept. That is, data lakes were still relatively new and ill-mastered at the beginning of this thesis. To clarify this, we propose a new definition of data lakes, as well as an analysis of approaches to metadata management and architectural organization in data lakes.

Based on an extensive state of the art, we identify the strengths and the limitations of existing approaches to metadata management in data lakes, which highlight that most of metadata management approaches are too specific to be reused. The only generic approaches are also limited either in terms of data types or in terms of supported functionalities. Thus, we address these shortcomings by introducing two metadata models called MEDAL and goldMEDAL.

Finally, we address the problems related to the effective implementation of data lakes, by proposing a data lake implementation called AUDAL that supports textual and tabular documents. This system is based on MEDAL and provides a set of extensible analysis services suitable for business users. To evaluate AUDAL, we propose and applied a benchmark dedicated to the quantitative evaluation of performance in data lakes, called DLBench. Through this quantitative evaluation and another qualitative evaluation (user experience), we demonstrate the effectiveness and usability of AUDAL.

Keywords : Decision-support systems, Data lakes, Metadata management, Metadata modeling, Benchmarking

Résumé

Au cours de la dernière décennie, le concept de lac de données (*data lake*) a émergé comme une alternative aux entrepôts de données pour le stockage et l'analyse des mégadonnées (*big data*). Les lacs de données adoptent une approche de stockage sans schéma fixe pour fournir un système d'aide à la décision souple et extensible. Concevoir un lac de données requiert avant tout de mettre en place un système de métadonnées efficace. En l'absence d'un schema fixe de données, les métadonnées sont en effet essentielles pour supporter les analyses et empêcher ainsi le lac de se transformer en marécage de données (*data swamp*), c'est-à-dire un lac de données inutilisable.

Si la littérature semble unanime sur l'importance du système de métadonnées, des interrogations et des incertitudes subsistent toutefois sur la méthodologie à suivre pour le mettre en œuvre. Plusieurs approches ont été proposées pour organiser les métadonnées dans les lacs de données, mais la plupart d'entre elles ne supportent pas d'analyses industrialisées comme dans les entrepôts de données. Par ailleurs, une part non négligeable de la littérature limite l'accès au lac de données aux seuls spécialistes du traitement de données (*data scientists*), excluant ainsi les experts métiers. De plus, la grande majorité des approches existantes d'organisation des métadonnées dans les lacs de données concerne uniquement les données structurées et semi-structurées. Concevoir un système de métadonnées supportant à la fois des analyses industrialisées et des données non structurées est donc encore une question de recherche ouverte.

C'est dans ce contexte que nous proposons à travers cette thèse un ensemble de contributions à la littérature sur la conception et la mise en œuvre de lacs de données. Nos contributions se déclinent en trois axes. Le premier axe se consacre à la désambiguïsation du concept de lac de données. Les lacs de données étaient en effet encore relativement nouveaux et mal maîtrisés au début de cette thèse. Pour remédier à cela, nous avons proposé une nouvelle définition des lacs de données, ainsi qu'une analyse des approches de gestion des métadonnées et d'organisation architecturales dans les lacs de données.

Un travail exhaustif d'état de l'art nous a permis d'identifier les forces et, surtout, les limites des approches existantes d'organisation des métadonnées dans les lacs de données. La plupart des approches sont en effet spécifiques à des cas d'usage précis et donc difficilement réutilisables. Les seules approches génériques sont elles aussi limitées non seulement par rapport aux types de données pris en charge, mais aussi en termes de fonctionnalités supportées. Nous remédions à ces insuffisances en introduisant deux modèles de métadonnées nommés MEDAL et goldMEDAL.

Nous avons enfin abordé les problématiques liées à la mise en œuvre effective de lacs de

données. Pour ce faire, nous avons proposé une implémentation de lac de données intitulée AUDAL, qui supporte des documents textuels et tabulaires. Ce système basé sur le modèle MEDAL propose un ensemble de services d'analyses extensibles adaptés aux utilisateurs métiers. Pour évaluer AUDAL, nous proposons et mettons en œuvre un banc d'essais dédié à l'évaluation quantitative des performances des lacs de données, nommé DLBench. Cette évaluation quantitative, complétée par une évaluation qualitative (expérience-utilisateur) démontrent l'efficacité et l'utilisabilité d'AUDAL.

Mots clés : Systèmes d'information décisionnels, Lacs de données, Gestion des métadonnées, Modélisation des métadonnées, Bancs d'essais

Remerciements

D'après l'écrivain africain Hampaté Bâ, « quelle que soit la valeur du présent fait à un homme, il n'y a qu'un mot pour témoigner la reconnaissance inspirée par la libéralité, et ce mot c'est : merci. » (L'Etrange destin de Wangrin, page 47). Ainsi voudrais-je au début de ce manuscrit exprimer ma reconnaissance à toutes les personnes qui d'une manière ou d'une autre ont contribué à la réalisation de cette thèse, et plus globalement à la construction du doctorant que je suis. A toutes ces personnes, je dis merci dans toutes les langues que je connais (et même dans celles que je ne connais pas) : *thank you*, *danke*, *gracias*, *grazie*, *barka* (en mooré), *anitché* (en bambara)...

Je voudrais tout d'abord rendre grâce au Très-Haut, qui est la source et la fin de toute chose. C'est avant tout à Toi que je dois le parcours qui m'a conduit à réaliser cette thèse. Tu as été pour moi une lumière sur mon chemin, une assurance dans mes moments de doutes, un abris quand j'étais dans la tourmente et un réconfort face au chagrin.

Je remercie tous mes enseignants du primaire, secondaire et du supérieur. Vos enseignements ont créé et attisé en moi une curiosité qui m'a prédisposé au monde de la recherche scientifique. Vous avez par ailleurs suscité en moi – à travers votre dévouement et vos manies respectives – une volonté de vous ressembler. Je remercie plus particulièrement YAYA TRAORE qui le premier (en 2014) m'a insufflé l'idée du parcours doctoral. Tu m'as fait prendre science en mon potentiel scientifique. Je remercie également RICCO RAKOTOMALALA de m'avoir accepté dans sa formation, et surtout de m'avoir accompagné dans la préparation de mon parcours doctoral.

J'adresse mes chaleureux et sincères remerciements à JÉRÔME DARMONT qui m'a fait confiance et qui m'a mis dans les meilleures dispositions pour réaliser cette thèse. Je te suis reconnaissant pour ta disponibilité, pour tes nombreux conseils et pour les inombrables heures de relectures. Je suis particulièrement admiratif de ta constante bienveillance et de ta simplicité. J'espère humblement pouvoir imiter ces qualités dans ma future carrière.

Merci par dessus tout d'avoir guidé mes premiers pas dans le monde de la recherche scientifique.

Je traduits toute ma reconnaissance aux rapporteurs, et autres membres du jury pour le temps consacré à la lecture et à l'évaluation de mon travail. Je remercie particulièrement ANNE LAURENT qui été membre de mon comité de suivi individuel de thèse. A travers tes questions, remarques et suggestions, tu m'as apporté un regard extérieur qui m'a aidé à prendre plus de recul. Je tiens également à mentionner FRANCK RAVAT qui a accepté (avec ANNE LAURENT) d'évaluer mon travail.

Je dis merci (par ordre alphabétique) à JAVIER ESPINOSA-OVIEDO, CÉCILE FAVRE, PENGFEI LIU, SABINE LOUDCHER et ETIENNE SCHOLLY – membres du groupe des « data lakers » du laboratoire ERIC – avec qui j'ai travaillé, co-écrit et *cogité* à travers d'inombrables séances de travail. De façon plus générale, je remercie tous les membres du laboratoire ERIC, avec qui et auprès de qui j'ai énormément appris au cours de ces trois ans de thèse.

Enfin, j'adresse mes remerciements aux chercheuses du laboratoire COACTIS avec qui j'ai collaboré dans le cadre du projet AURA-PMI. Merci donc à MARTINE SEVILLE, ISABELLE PRIM-ALLAZ et TININHANE TAZAIR. Je ne saurais finir sans mentionner le Région Auvergne-Rhône-Alpes, qui a financé le projet AURA-PMI, et à travers ce projet, a donc financé ma thèse.

Dédicace

Pourquoi je fais une thèse ? Cette question m'a souvent été posée, et j'ai presque toujours hésité entre plusieurs raisons. Je pourrais dire que c'est pour la fierté d'atteindre le dernier grade universitaire. Ce ne serait pas faux, mais ce ne serait pas totalement vrai non plus. La principale raison est que la thèse était à mes yeux un moyen d'embrasser le métier d'enseignant-chercheur. On pourrait me demander alors pourquoi je souhaite devenir enseignant-chercheur ?

Ce désir, ou plutôt cette vocation, je la dois à mes géniteurs, tous deux instituteurs. Le métier d'enseignant-chercheur serait pour moi un moyen de les imiter. Mais avant de les imiter dans une future carrière, je voudrais déjà leur rendre hommage en leur dédiant cette thèse. Merci Papa, merci Maman.

Merci Papa pour ton exemple. Tu m'as donné le goût de la connaissance et du savoir. C'est aussi cette volonté de « comprendre comment les choses fonctionnent » qui m'a conduit jusqu'au parcours doctoral. Tu m'as aussi, et surtout appris à être polyvalent. C'est cela qui m'a conduit à faire un BAC D, et non pas le BAC A littéraire auquel je me prédestinais ; et qui m'aurait sans-doute conduit à une autre voie.

Merci Maman pour ton exemple. Je me souviens de ces cours-à-domicile où tu m'apprenais à lire, à écrire, à calculer, à raconter, à organiser, et même à chanter. Ce sont ces « petites choses » qui m'ont permis plus tard de remporter un concours de dissertation littéraire, et aujourd'hui de (co)-rédiger des articles et une thèse. Merci par dessus tout de m'avoir transmis la Foi, lumière de ma vie. Ton départ a créé en moi un grand vide. Mais nul ne meurt tant que son œuvre perdure. Tu vivras toujours à travers tes enfants et leurs œuvres, et à travers ces milliers d'élèves que tu as formé et éduqué.

Je dédie également ce travail à mes frères et sœurs qui ont toujours été pour moi une source de motivation et de reconfort. Merci à JULIE, à THÈRÈSE, à BÉNOÎT, à RENAUD, à PATRICE et à PATRICIA ; merci d'être là.

Table des matières

Chapitre 1

Introduction générale

1.1 Contexte

Depuis le début du XXI[e] siècle, nous assistons à une croissance exponentielle de la production de données dans le monde. Ces mégadonnées, plus couramment appelées *big data*, proviennent notamment des systèmes d'information d'entreprises et plus encore des médias sociaux (Facebook, Twitter, Wikipédia, Youtube, etc.) et des objets connectés (*Internet of Things*). Elles représentent une opportunité indéniable pour les organisations. En effet, l'exploitation des mégadonnées procure un avantage concurrentiel important en offrant aux entreprises des indicateurs pour éclairer leurs décisions, et ainsi améliorer leurs performances (Beheshti et al., 2018).

Cependant, tirer profit des mégadonnées nécessite de remédier aux problématiques qu'elles posent. En effet, les *big data* sont caractérisées par un grand volume, une extrême vélocité et une grande variété qui rendent complexes leur gestion et leur exploitation (Ajah & Nweke, 2019 ; Miloslavskaya & Tolstoy, 2016). Les entreprises sont donc à la recherche de solutions et d'outils toujours plus innovants pour répondre aux défis liés à ces caractéristiques. Cette quête va croissante et s'accélère même. Pour preuve, le marché mondial de la gestion et de l'analyse des mégadonnées devrait passer de 193 milliards de dollars américains en 2019 à 421 milliards en 2027 (Borasi et al., 2020).

En guise de solution aux besoins de gestion et d'analyse des mégadonnées, le concept d'entrepôt de données (*data warehouse*) a été largement utilisé au cours des trois dernières décennies. Il s'agit d'un système de stockage et d'organisation des données permettant d'en tirer de la valeur à travers des analyses industrialisées. Cependant, si les entrepôts de données se sont montrés performants pour les données structurées, ils se sont en revanche révélés très limités en ce qui concerne les données semi-structurées et non structurées. L'entrepôt de données utilise en effet un schema fixe et prédéfini auquel les données doivent se conformer. On parle d'approche *schema-on-write* (Khine & Wang, 2017). Cette approche convient peu aux données semi-structurées et surtout non structurées, puisque ces données ont par définition un schéma plus ou moins inexistant. De plus, l'approche *schema-on-write* propre aux entrepôts de données impose une rigidité qui limite les possibilités d'analyses (LaPlante & Sharma, 2016).

Les entrepôts de données étant ainsi incapables de répondre intégralement aux défis que représentent le stockage et l'exploitation des mégadonnées, un nouveau concept a été proposé par Dixon (2010) : le lac de données (*data lake*). Il s'agit d'un système visant à stocker, traiter et rendre exploitables des données de toutes structures (données structurées, semi-structurées ou non structurées) et de formats quelconques. Le concept de lac de données est longtemps resté ambigu. Ainsi, il fut d'abord considéré comme un simple concept marketing (Grosser et al., 2016), ou assimilé à la technologie Hadoop (Fang, 2015 ; O'Leary, 2014).

Nonobstant le scepticisme initial, les avantages associés à ce nouveau concept ont fini par susciter l'intérêt aussi bien en milieu académique qu'industriel. En effet, les lacs de données permettent un stockage des données 10 à 100 fois moins coûteux que dans les systèmes traditionnels (Stein & Morrison, 2014). Il offre surtout une flexibilité lui permettant de supporter une plus grande diversité d'analyses que l'entrepôt de données (LaPlante & Sharma, 2016). Ainsi, outre la prise en charge de données semi-structurées et non

structurées, le lac de données permet des analyses croisées de données hétérogènes et évite donc le silotage des données (Wibowo et al., 2017). Cependant, le concept de lac de données reste en cours de maturation (Russom, 2017). Par conséquent, plusieurs visions du lac de données se confrontent, aussi bien concernant sa conception que son exploitation.

1.2 Problématiques et objectifs

À travers cette thèse, nous travaillons à remédier à deux problématiques distinctes, mais intimement liées. D'une part, notre travail répond à un besoin métier (Section 1.2.1). Cependant, la résolution de ce besoin métier nécessite de remédier à une deuxième problématique d'ordre scientifique (Section 1.2.2).

1.2.1 Problématique métier

Cette thèse se déroule dans le cadre du projet AURA-PMI. Il s'agit d'un projet de recherche pluridisciplinaire incluant deux équipes de recherche en informatique et en sciences de la gestion. Le projet AURA-PMI vise à étudier, comparer et analyser les stratégies de digitalisation et de servicisation des petites et moyennes entreprises industrielles dans la région Auvergne-Rhône-Alpes. Pour ce faire, un ensemble de données essentiellement textuelles a été recueilli auprès des entreprises cibles, dans l'idée d'en extraire des tendances, ainsi que des ressemblances ou dissemblances entre les entreprises.

Le jeu de données à stocker et à analyser dans le cadre du projet AURA-PMI est composé de 8 200 documents textuels d'une part et de 6 fichiers tabulaires d'autre part. Les documents textuels sont de plusieurs types (rapports financiers, rapports d'activités, communiqués de presse, etc.), en deux langues (français, anglais) et d'une longueur moyenne de 12 pages. Les fichiers tabulaires contiennent chacun des informations détaillées (localisation, secteur d'activité, stratégie digitale, etc.) ou des indicateurs numériques sur les entreprises (effectifs, cours moyens en bourse, indices de digitalisation, proportion de servicisation, etc.). La figure 1.1 illustre les caractéristiques des données traitées dans le cadre du projet AURA-PMI.

Si ces données ne posent pas de véritable problème de stockage à cause de leur volume limité (6,2 Go), elles induisent en revanche des besoins spécifiques pour leur analyse. En effet, un système informatique s'avère indispensable pour supporter l'analyse d'une telle quantité de données. Pour répondre aux besoins du projet AURA-PMI, un tel outil doit prendre en compte plusieurs contraintes.

D'abord, l'outil à mettre en place doit offrir une solution unifiée permettant de stocker et d'analyser à la fois les documents textuels et fichiers tabulaires dont dispose le projet AURA-PMI. Ensuite, il doit offrir une large panoplie d'analyses avancées, permettant notamment de résumer, décrire et comparer des groupes de documents, et ce de façon industrialisée. En effet, les analyses étant réalisées par des utilisateurs métiers, il est indispensable de leur proposer une solution autonome et ne nécessitant donc pas de compétences particulières en traitements de données. Enfin, l'outil de stockage et d'analyse souhaité doit être extensible, de sorte à supporter plus tard des axes et des types d'analyses

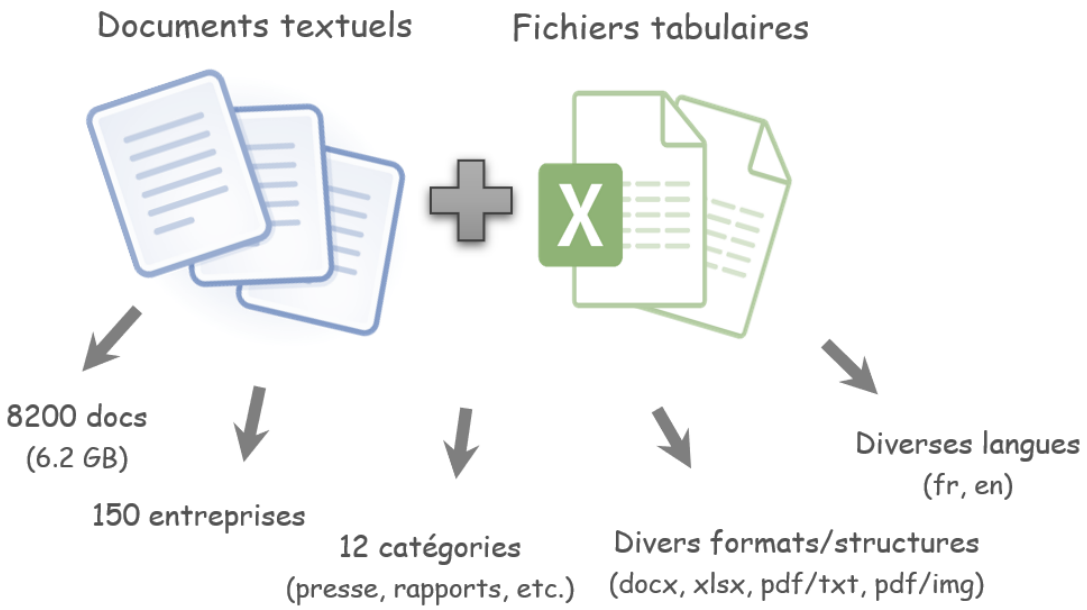


FIGURE 1.1 – Caractéristiques des données du projet AURA-PMI

non encore déterminés.

Le concept à mesure de répondre au mieux aux contraintes précitées est le lac de données. En effet, le lac de données gère aussi bien des données structurées, semi-structurées, que non structurées, et offre une très large panoplie d'analyses. Cependant, il reste à déterminer comment concevoir et construire un tel lac de données.

1.2.2 Problématique de recherche

Concevoir un lac de données requiert avant tout de mettre en place un système de métadonnées efficace. En l'absence d'un schema fixe de données (*schema-on-read*), les métadonnées sont en effet essentielles pour rendre possibles les analyses et empêcher ainsi le lac de se transformer en marécage de données (*data swamp*), c'est-à-dire en un lac de données inutilisable (ALREHAMY & WALKER, 2015 ; HAI et al., 2016 ; SURIARACHCHI & PLALE, 2016).

Si la littérature semble unanime sur l'importance du système de métadonnées, des interrogations et des incertitudes subsistent toutefois sur la méthodologie à suivre pour le mettre en place. Plusieurs approches ont été proposées pour organiser les métadonnées dans les lacs de données, mais la plupart d'entre elles ne supportent pas d'analyses industrialisées comme dans les entrepôts de données. D'ailleurs, une part non-négligeable de la littérature limite l'accès au lac de données aux seuls spécialistes du traitement de données (statisticiens, *data scientists*, *data analysts*), excluant ainsi les utilisateurs métiers (KHINE & WANG, 2017 ; MADERA & LAURENT, 2016 ; MATHIS, 2017 ; SIROSH, 2016).

De plus, la grande majorité des approches existantes d'organisation des métadonnées dans les lacs de données concerne uniquement les données structurées et semi-structurées (FARID et al., 2016 ; HAI et al., 2016 ; MACCIONI & TORLONE, 2018 ; QUIX et al., 2016), alors

que la majorité des données à disposition des entreprises est non structurée, et donc très souvent textuelle (Ajah & Nweke, 2019 ; Miloslavskaya & Tolstoy, 2016). Concevoir un système de métadonnées supportant à la fois des analyses industrialisées et des données non structurées est donc encore une question de recherche ouverte.

Enfin, la spécificité du projet AURA-PMI nécessite d'aller au delà des technologies communément citées pour la mise en œuvre de lacs de données. En effet, le concept de lac de données est souvent associé et même confondu à la technologie Apache Hadoop (Couto et al., 2019 ; Fang, 2015 ; Ganore, 2015 ; O'Leary, 2014). Pourtant, l'utilisation de l'écosystème Apache Hadoop nécessite des ressources matérielles et logicielles, ainsi qu'une expertise dont le coût est incompatible avec les capacités du projet AURA-PMI en particulier, et plus largement des petites et moyennes entreprises en général. Par conséquent, des technologies alternatives à Apache Hadoop sont nécessaires.

1.2.3 Objectifs

En lien avec les problématiques métier et scientifique définies précédemment, l'objectif de cette thèse est de contribuer à la littérature à travers la **conception et la mise en œuvre d'un lac de données supportant des analyses assistées de documents textuelles et tabulaires**. Cet objectif global peut se décliner en trois parties.

La première étape consiste à proposer un modèle d'organisation des metadonnées dans les lacs de données. À travers ce modèle, nous visons à proposer une méthodologie d'organisation des métadonnées, permettant de transcender les limites des approches existantes. Le modèle à proposer devrait donc être suffisamment générique pour traiter à la fois des données structurées, semi-structurées et non structurées.

Une fois ce modèle de métadonnées effectif, nous pouvons alors nous en servir pour concevoir un lac de données prenant en charge des documents textuels, d'une part, et des fichiers tabulaires, d'autre part. Plus concrètement, il s'agit de proposer une implémentation du modèle introduit précédemment en identifiant et en mettant en œuvre les technologies, traitements et fonctionnalités adéquats.

Enfin, la relative nouveauté du concept de lac de données fait qu'il n'existe pas encore de standards d'évaluation des implémentations de lacs de données. Par conséquent, nous devrons dans un troisième temps proposer une méthodologie permettant d'évaluer notre implémentation. Un tel banc d'essais (*benchmark*) devra alors contenir un modèle de données, un modèle de charge et un ensemble de métriques permettant de mettre en exergue les forces et faiblesses d'un lac de données dans un contexte de données tabulaires et textuelles.

1.3 Organisation du manuscrit

La suite de ce manuscrit est organisée en trois parties. La première partie présente un état de l'art sur les lacs de données. Elle est composé de trois chapitre.

— Dans le **chapitre 2** nous présentons un historique des systèmes décisionnels. Pour ce faire, nous partons du concept d'entrepôt de données, traditionnellement utilisé

pour l'aide à la décision. Nous en soulignons les limites et montrons comment les lacs de données s'en démarquent.

— Dans le **chapitre 3**, nous nous intéressons à l'organisation des métadonnées, qui est une problématique clé associée à la conception des lacs de données. Nous y proposons une analyse comparative des approches existantes d'organisation des métadonnées dans les lacs de données.

— Dans le **chapitre 4**, nous présentons les architectures, ainsi que les technologies et techniques utilisées et utilisables pour la mise en œuvre des lacs de données. Nous y proposons notamment une typologie des principales approches architecturales d'organisation des lacs de données.

La deuxième partie de ce mémoire présente nos contributions en matière de modélisation et d'organisation des métadonnées dans les lacs de données. Nous y présentons deux modèles de métadonnées que nous avons proposés.

— Dans le **chapitre 5**, nous introduisons notre premier modèle de métadonnées nommé MEDAL. À travers MEDAL, nous utilisons la théorie des graphes et adoptons une vision étendue des métadonnées pour gérer efficacement et de façon adéquate les interactions entre les données et les métadonnées dans le lac de données.

— Dans le **chapitre 6**, nous présentons goldMEDAL, une évolution de MEDAL. Le modèle goldMEDAL adopte un plus haut niveau d'abstraction pour représenter les métadonnées du lac, ce qui le rend plus simple et plus générique.

La troisième partie du manuscrit est consacrée à nos propositions en matière d'implémentation et d'évaluation de lacs de données. Elle comprend trois chapitres.

— Dans le **chapitre 7**, nous introduisons le lac de données AUDAL, resultat de la mise en œuvre du modèle MEDAL. Nous y présentons les choix techniques et technologiques ayant concouru à la conception du lac de données AUDAL.

— Dans le **chapitre 8**, nous introduisons DLBench, un banc d'essais dédié à l'évaluation de lacs de données. Nous y proposons un jeu de données, un ensemble de requêtes et de métriques, ainsi qu'un protocole d'évaluation tous adaptés à un contexte de données textuelles et tabulaires.

— Dans le **chapitre 9**, nous évaluons le lac de données AUDAL. Pour ce faire, nous mesurons les performances de notre implémentation à l'aide de DLBench. Nous complétons cette évaluation quantitative par une évaluation qualitative de l'expérience-utilisateurs, ainsi qu'à travers une confrontation avec les critères de référence proposés dans la littérature.

Enfin, nous concluons ce manuscrit dans le **chapitre 10**, qui fait le bilan de nos différentes contributions, d'une part, et présente les questions de recherche qui demeurent, d'autre part.

Première partie

État de l'art

Chapitre 2

Des entrepôts de données aux lacs de données

Sommaire

2.1 Introduction

Le concept de lac de données auquel nous nous intéressons dans cette thèse a été défini pour répondre aux insuffisances des entrepôts de données (Dixon, 2010). Les lacs de données sont d'ailleurs présentés comme une approche moderne de stockage et d'exploitation des données, en opposition à l'approche traditionnelle d'entreposage de données (Laskowski, 2016 ; Singh et al., 2016 ; Suriarachchi & Plale, 2016). On peut donc appréhender le concept de lac de données en le comparant à son prédécesseur, l'entrepôt de données.

Ainsi, nous présentons dans ce chapitre une analyse du processus ayant conduit à passer des entrepôts aux lacs de données pour le stockage et l'analyse des données. Nous commençons par présenter le concept d'entrepôt de données à travers son historique, ses objectifs et son fonctionnement. Ensuite, nous présentons les insuffisances des entrepôts de données, ainsi que plusieurs variantes d'entrepôts proposées pour y faire face. Enfin, nous présentons le concept de lac de données qui introduit un nouveau paradigme de stockage et d'analyse des données.

2.2 Systèmes décisionnels traditionnels

2.2.1 Du modèle relationnel au modèle multidimensionnel

En 1970, E.F. Codd a introduit le modèle relationnel, un nouveau format de stockage et de représentation des données basé sur l'algèbre relationnelle (Codd, 1970). Ce nouveau modèle a alors révolutionné le stockage des données en remédiant aux insuffisances des approches précédentes. En effet, les systèmes de gestion de base de données (SGBD) étaient traditionnellement conçus sur une organisation hiérarchique ou en réseau des données, dont le défaut principal était un manque d'indépendance des données. Les bases de données suivant ce format étaient alors difficiles et coûteuses à mettre en œuvre, à maintenir et à utiliser (Codd et al., 1993). À l'inverse, les SGBD relationnels offraient une facilité de mise en œuvre et d'utilisation qui en ont fait une référence dans les systèmes d'information d'entreprises.

Au cours des deux décennies suivantes, la facilité offerte par les SGBD relationnels et la croissance des systèmes d'information d'entreprises ont engendré un besoin accru d'analyses de données. En effet, l'analyse de l'activité des entreprises leur permet d'orienter au mieux leurs décisions et leur procure ainsi un avantage concurrentiel non négligeable sur leurs concurrents (Teste, 2000). L'analyse de données est alors devenue de plus en plus systématique et l'on tendait ainsi vers une industrialisation des analyses pour l'aide à la décision au sein des entreprises.

Cependant, l'organisation relationnelle des données, éprouvée pour le traitement de requêtes transactionnelles, s'est vite montrée limitée face à ce nouveau besoin. Pour y remédier, Codd et al. (1993) proposent un tout autre modèle dans lequel les données sont organisées suivant les futurs besoins d'analyses : le modèle multidimensionnel. Codd et al. (1993) distinguent ainsi d'une part les systèmes OLTP (*On-Line Transactionnal Processing*) qui supportent les données de production suivant une approche relationnelle classique et, d'autre part, les systèmes OLAP (*On-Line Analytical Processing*) qui suivent un modèle multidimensionnel en prévision d'analyses futures (Codd et al., 1993).

2.2.2 Systèmes OLAP

12 règles de Codd

En introduisant le modèle multidimensionnel, Codd et al. (1993) définit douze règles que les outils OLAP, c'est-à-dire les outils implémentant le modèle multidimensionnel, doivent suivre.

1. **Conception multidimensionnelle :** Le système doit suivre un modèle multidimensionnel.
2. **Transparence :** Le système doit permettre aux utilisateurs de réaliser les analyses de façon aisée et efficace
3. **Accessibilité :** L'outil OLAP doit fournir une vision unique cohérente et compréhensible des données pour être facilement pris en main par les utilisateurs
4. **Performances :** Le système doit assurer et maintenir un haut niveau de performances, indépendamment du nombre de dimensions.
5. **Client-Serveur :** Le système doit suivre une architecture 2-tiers (ou 3-tiers) permettant ainsi de fournir les analyses de façon uniforme depuis le serveur.
6. **Dimensions génériques :** Toutes les dimensions doivent être structurées et utilisées de façon uniforme.
7. **Gestion de l'éparsité :** Le système doit gérer de façon efficiente l'utilisation de la mémoire, particulièrement en présence de données éparses
8. **Multi-utilisateurs :** L'outil OLAP doit permettre des accès concurrent, tout en garantissant l'intégrité des données ainsi que leur sécurité.
9. **Croisement de dimensions :** Le système doit permettre de croiser un nombre illimité de dimensions lors des analyses.
10. **Manipulation intuitive des données :** L'utilisation de l'outil OLAP doit être naturel, particulièrement pour les utilisateurs métiers.
11. **Flexibilité :** Le système doit permettre de présenter les données sous plusieurs perspectives.
12. **Dimensions et agrégations illimitées :** L'outil OLAP doit supporter un grand nombre de dimensions et d'agrégations concurrentes.

Systèmes OLAP vs. Systèmes OLTP

Les systèmes OLAP ainsi définis se distinguent des systèmes transactionnels sur plusieurs aspects, notamment en ce qui concerne leurs utilisations et la nature des données traitées (Favre, 2007 ; Teste, 2000). Ainsi, les données sont détaillées et organisées suivant des activités dans les systèmes OLTP. *A contrario*, les systèmes OLAP recueillent des données résumées et organisées suivant des sujets d'analyse. Par ailleurs, les données des systèmes OLTP évoluent et se volatilisent au fil des opérations. Cela leur confère une taille moins importante que dans les systèmes OLAP, où les données sont historisées.

Du point de vue des utilisations, les systèmes OLTP supportent de nombreux accès concurrents pour des opérations de mise à jour et d'interrogation. Les systèmes OLAP ont quant à eux un nombre et une panoplie plus limités de requêtes, mais qui sont plus complexes.

TABLE 2.1 – Comparaison des systèmes OLTP et OLAP

Système → **Critère ↓**	**Système OLTP**	**Système OLAP**
Données	Exhaustives, détaillées	Agrégées, résumées
	Actuelles	Historiques
	Dynamique	Statique
	Orientées application	Orientées sujet
	Plusieurs giga-octets	Plusieurs tera-octets
Utilisateurs	Nombreux	Peu nombreux
	Mise à jour & interrogations	Interrogations
	Requêtes prédéfinies & simples	Requêtes imprévisibles & complexes
	Accès concurrents	Accès non concurrents
	Réponses rapides	Réponses moins rapides

Par conséquent, les requêtes sont plus coûteuses en temps dans les systèmes OLAP. Dans la Table 2.1, nous présentons plus en détails les différences entre les systèmes OLAP et OLTP.

2.2.3 Entrepôts et magasins de données

Au début des années 1990, les systèmes OLAP proposés par CODD et al. (1993) ont donné lieu à l'émergence de systèmes ou d'architectures décisionnels au sein des entreprises. Les architectures décisionnelles regroupent un ensemble d'informations et d'outils organisés de sorte à supporter de manière efficace la prise de décision (MADERA, 2018 ; TESTE, 2000). Plus formellement, nous nous référons dans ce document aux architectures décisionnelles en suivant la définition de MADERA (2018).

Définition 2.2.1. Une architecture décisionnelle (ou système décisionnel) est une partie du système d'information dédiée aux applications décisionnelles.

Les architectures décisionnelles sont classiquement composées de quatre éléments essentiels (KIMBALL & ROSS, 2013) : les sources de données, l'entrepôt de données, les magasins de données et les outils d'analyse (W. H. INMON, 1994 ; KIMBALL & ROSS, 2013). On y ajoute aussi de façon optionnelle un cinquième élément, le réceptacle ou plus communément l'*Operational Data Store* (ODS) (Figure 2.1).

Sources de données

Une première partie des données intégrées dans l'architecture décisionnelle provient de sources internes à l'entreprise. Il s'agit essentiellement des bases de données de productions, c'est à dire des systèmes OLTP. Une deuxième partie des données traitées provient de sources externes. Il s'agit notamment de données gouvernementales, de données d'entreprises partenaires, etc.

ODS

C'est un composant *optionnel* du système décisionnel. Il peut être vu comme un sas de préparation dans lequel les données sont intégrées et préparées pour alimenter l'entrepôt de données et parfois les magasins de données (W. H. Inmon, 2005). Les données y sont volatiles, c'est à dire que seules les valeurs actuelles sont conservées. Cela permet à l'ODS de supporter des analyses devant être réalisées en temps réel.

Entrepôt de données

C'est l'élément central de l'architecture décisionnelle. Il entre dans la catégorie des systèmes OLAP tels que définis par Codd et al. (1993). Le concept d'entrepôt de données a été introduit et vulgarisé par W. H. Inmon (1994) et Kimball (1996), qui en donnent une définition consensuelle.

Définition 2.2.2. Un entrepôt de données est une collection de données intégrées, orientées sujet, non volatiles, historisées, résumées et disponibles pour l'interrogation et l'analyse

De cette définition, on note six caractéristiques des entrepôts de données.

- **Données intégrées** : dans l'entrepôt, les données sont présentées de façon harmonisée. Cela permet de remédier à l'hétérogénéité des formats de présentation, induits par la multiplicité des sources de données. Par exemple, la multiplicité de formats possibles de présentation du sexe dans les données sources (homme/femme, h/f, masculin/feminin, m/f, 1/0, etc.) se traduit par un seul format dans l'entrepôt.
- **Données orientées sujet** : l'entrepôt de données fait abstraction de l'organisation fonctionnelle des données atypique aux systèmes opérationnels. À la place, les données sont organisées en sujets d'analyse.
- **Données non volatiles** : l'entrepôt de données est dédié et limité à des opérations de lecture. Il n'y a donc plus de modification des données, une fois qu'elles sont stockées dans l'entrepôt.
- **Données historisées** : pour assurer la non-volatilité des données tout en intégrant les changements dans l'entrepôt de données, un système d'historisation est utilisé. Ainsi, l'entrepôt de données conserve à la fois les valeurs actuelles et anciennes des données.
- **Données résumées** : les données intégrées dans l'entrepôt sont agrégées au plus haut niveau de granularité, de manière à supporter l'ensemble des analyses futures. Par exemple, des ventes seraient agrégées par jour si l'on souhaite se limiter à une analyse de l'activité par jour, semaine, mois, etc.

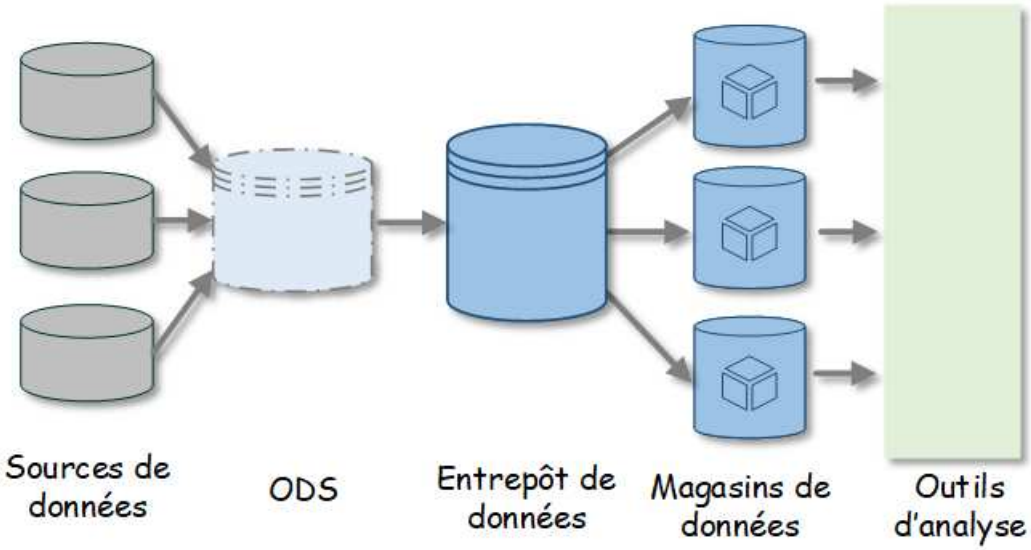


FIGURE 2.1 – Élements d'une architecture décisionnelle

— **Données disponibles pour l'analyse** : un accès simple et approprié est accordé aux utilisateurs pour interroger et analyser les données dans l'entrepôt.

Magasin de données

Le concept de magasin de données est étroitement lié à celui d'entrepôt de données. Il peut aussi être considéré comme un système OLAP. En suivant la vision de KIMBALL (1996), il peut être définit comme suit.

Définition 2.2.3. Un magasin de données est un sous-ensemble d'un entrepôt de données dédié à un ensemble spécifique d'analyses.

Un magasin de données répond le plus souvent aux besoins d'analyses d'un département précis. Ainsi, l'architecture décisionnelle peut inclure, par exemple, un magasin de données dédié aux analyses liées au département des ressources humaines, un autre pour la compatibilité, etc.

Outils d'analyse

Ils permettent aux utilisateurs d'interagir avec l'architecture décisionnelle pour réaliser des analyses. Ces analyses sont faites soit depuis l'entrepôt de données, soit à partir d'un magasin de données. Certaines analyses avancées sont réalisées directement à partir de l'ODS. Cela nécessite toutefois une expertise et des outils particuliers (à cause de la complexité des traitements), contrairement aux analyses depuis l'entrepôt et les magasins qui, elles, sont réalisées par des utilisateurs métiers.

2.2.4 Conception d'entrepôts et de magasins de données

Les entrepôts et magasins de données répondent aux caractéristiques des systèmes OLAP tels que définis par CODD et al. (1993), dans le sens où ils suivent un modèle multidimen-

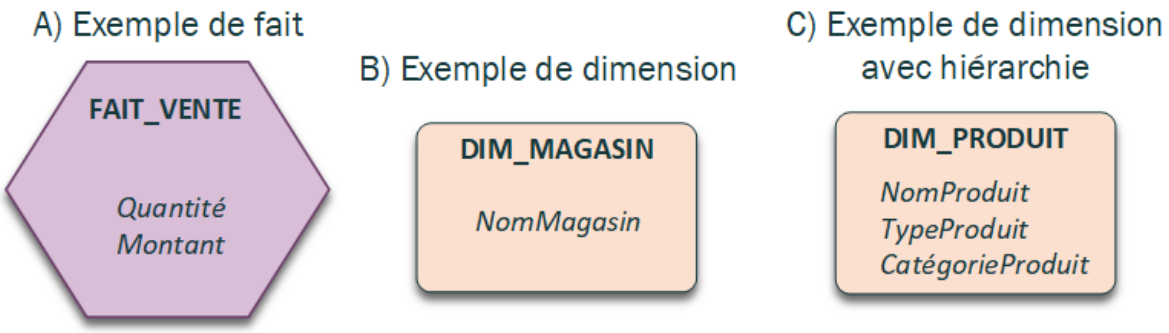


FIGURE 2.2 – Exemples de représentations de fait et de dimensions

sionnel qui consiste à organiser les données autour de perspectives d'analyses. Il existe maintenant des standards de conception de ces systèmes.

Concepts de base

La modélisation multidimensionnelle se base sur de multiples concepts.

Faits et mesures Le concept de fait représente un ensemble de mesures, reflétant l'activité de l'entreprise (KIMBALL & ROSS, 2013). Les mesures sont des valeurs très souvent quantitatives dont l'agrégation permet d'évaluer l'activité (Figure 2.2-A). Par exemple, des mesures appropriées pour analyser l'activité d'une chaîne de supermarchés pourraient être les *quantités vendues* (de produits) et les *montants encaissés*.

Les mesures associées au même fait doivent être au même niveau de granularité, c'est-à-dire au même niveau de détail. Par exemple, si l'on considère dans un fait la quantité vendue de chaque produit à chaque opération de vente, on doit également considérer le montant vendu par produit et par opération de vente. Si tel n'était pas le cas, il faudrait alors séparer les mesures dans des faits différents, sans quoi les futures analyses deviendraient incohérentes, voire erronées (KIMBALL & ROSS, 2013).

Dimensions et hiérarchies Le concept de dimension représente un ensemble de valeurs, très souvent catégorielles, qui décrivent l'activité analysée à travers le fait (KIMBALL & ROSS, 2013). Elles tracent le contexte (qui, quoi, où, quand, comment et pourquoi) de chaque mesure (Figure 2.2-B). Dans l'exemple de la chaîne de supermarchés évoqué précédemment, des dimensions pourraient représenter le *caissier* (qui), le *produit* vendu (quoi), le *site* (où), la *date* (quand), ainsi que le *type de paiement* à savoir en espèces, carte bancaire, bons d'achat, etc. (comment). Lors des analyses, les dimensions servent à filtrer les mesures et à les regrouper suivant des perspectives d'analyse.

Les valeurs des dimensions doivent être explicites et compréhensibles. Sont donc à éviter les codes et abréviations, ceci dans le but d'optimiser l'expressivité des résultats des futures analyses. Les dimension contiennent généralement plusieurs attributs, avec potentiellement des relations hiérarchiques entre elles. Par exemple, des produits peuvent être regroupés par types, eux-mêmes regroupés par catégories (Figure 2.2-C). Dans de tels

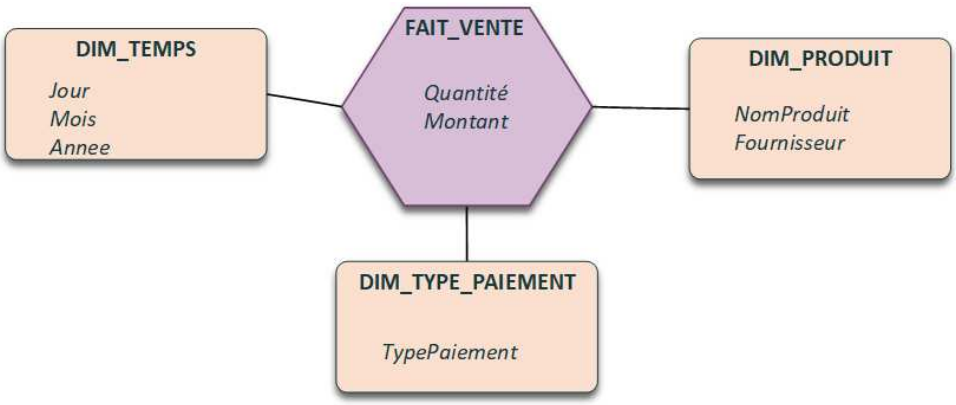


FIGURE 2.3 – Exemple de modèle en étoile

cas, ces attributs peuvent être conservés dans la même dimension (modèle en étoile) ou séparés (modèle en flocons de neige).

Modélisation conceptuelle

La modélisation conceptuelle permet d'identifier les différentes entités de données (dimensions et faits), ainsi que les relations qui existent entre elles (W. H. INMON, 2005). Il existe essentiellement trois modèles multidimensionnels au niveau conceptuel.

Modèle en étoile (*star schema*) En utilisant les concepts définis précédemment, l'activité de l'entreprise peut être représentée (dans le cas le plus simple) par un ensemble de dimensions toutes associées à un fait central (KIMBALL & ROSS, 2013). Cela donne une structure similaire à une étoile, d'où l'appellation « modèle en étoile ». Cette représentation offre l'avantage d'une grande simplicité. En contrepartie, elle ne permet pas d'expliciter clairement les relations hiérarchiques entre les dimensions.

> *Exemple :* Une pharmacie souhaite mettre en place un modèle multidimensionnel pour analyser les ventes de ses produits. Chaque produit est associé à un fournisseur. En plus de l'analyse temporelle et par fournisseur, la pharmacie souhaite tracer les types de paiement utilisés (tiers payant, carte bancaire, espèces, etc.). La Figure 2.3 présente le modèle en étoile résultant.

Modèle en flocons de neige (*snowflake schema*) Il ressemble en tous points à un modèle en étoile, sauf que les hiérarchies de dimensions sont explicitées dans le modèle en flocons de neige sous forme de dimensions. Le modèle en flocons de neige offre ainsi l'avantage d'une meilleure lisibilité. Mais cela engendre une complexification du modèle qui se voit à travers un plus grand nombre de dimensions.

> *Exemple :* La Figure 2.4 présente un modèle en flocons de neige, traduisant la même activité que le modèle en étoile présenté précédemment (Figure 2.3).

Modèle en constellation Il consiste à fusionner plusieurs modèles en étoile (ou en flocons de neige) à travers des dimensions communes (TESTE, 2000). Un modèle en constellation comprend donc au moins deux faits.

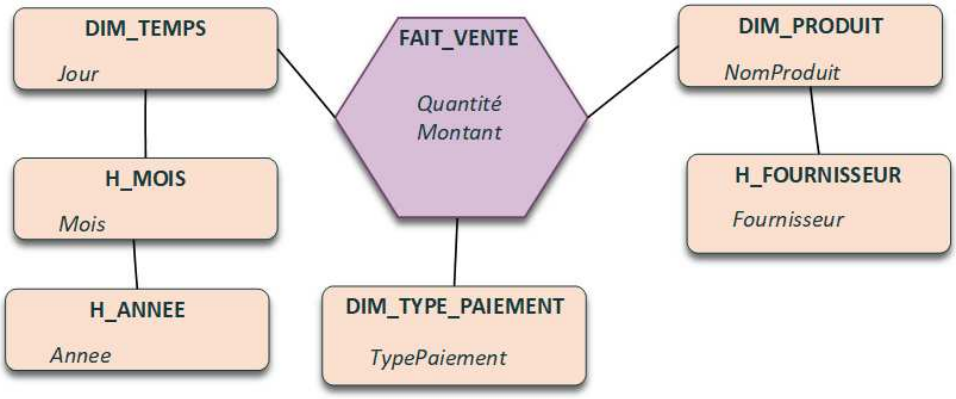


FIGURE 2.4 – Exemple de modèle en flocons de neige

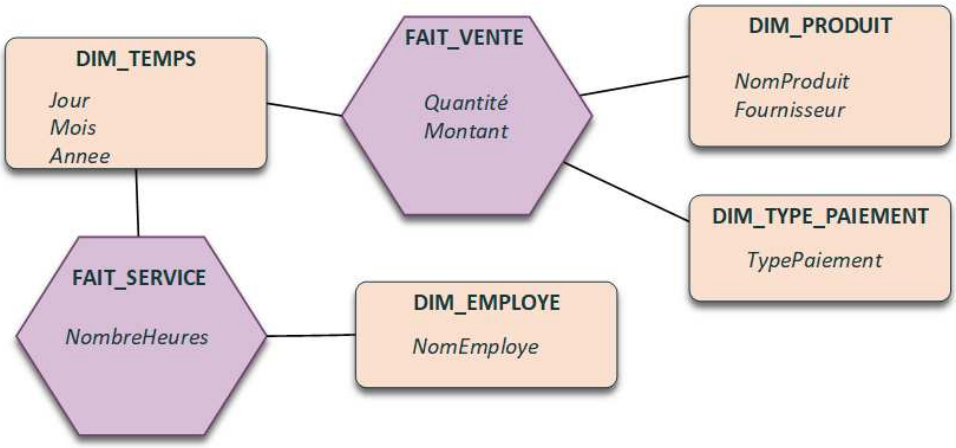


FIGURE 2.5 – Exemple de modèle en constellation

> *Exemple :* Imaginons à présent que le supermarché dont l'activité est représentée dans la Figure 2.3 souhaite aussi mesurer les heures de service de ses employés. En étendant le modèle initial, nous obtenons alors un modèle en constellation. Dans ce modèle, la dimensions *temps* est partagée par les deux faits *vente* et *service* (Figure 2.5).

Modélisation logique et physique

La modélisation logique consiste à définir le format effectif de stockage des données. Elle est étroitement liée au type de technologie utilisé (modèle physique). Les trois principaux modèles logiques utilisés en modélisation multidimensionnelle sont les modèles *Relational OLAP* (ROLAP), *Multidimensional OLAP* (MOLAP) et *Hybrid OLAP* (HOLAP).

ROLAP C'est l'approche la plus couramment utilisée. Elle consiste à implémenter le modèle conceptuel suivant une représentation relationnelle. Les dimensions et les faits sont alors traduits par des tables relationnelles. Ainsi la table de faits contient les mesures sous forme d'attributs, ainsi que les clés étrangères des tables de dimensions associées. Dans le modèle ROLAP, les analyses sont réalisées via le langage SQL, traditionnellement utilisé pour interroger les SGBD relationnels (CODD, 1970). Plus récemment, le lan-

gage MDX (*Multi-Dimensionnal Expressions*) a aussi été proposé pour l'OLAP (Langit, 2007).

MOLAP Ce modèle utilise nativement des structures multidimensionnelles pour le stockage des données. Ces structures sont connues sous l'appellation « tables multidimensionnelles », « cubes » ou « hypercubes » (Teste, 2000). En plus de fournir un stockage naturel des données, cette approche rend les analyses plus rapides et intuitives. Cependant, les modèles multidimensionnels passent difficilement à l'échelle, notamment à cause du problème d'éparsité des données (Favre, 2007).

HOLAP Ce modèle permet de tirer le meilleur de chacune des approches ROLAP et MOLAP. En effet, le modèle HOLAP consiste à adopter une vision relationnelle pour le stockage des données et une approche multidimensionnelle pour les analyses. Plus concrètement, les données détaillées sont stockées dans un SGBD relationnel à partir duquel les analyses sont effectuées via un moteur multidimensionnel. Cela permet de concilier à la fois un stockage optimal et des analyses rapides. L'approche HOLAP est cependant difficile à mettre en œuvre et les analyses sont moins rapides que dans un modèle MOLAP (Favre, 2007).

Le Tableau 2.2 fait la synthèse des différences entre les trois approches d'organisation logique et physique des données dans les outils OLAP.

Table 2.2 – Comparaison des modèles ROLAP, MOLAP et HOLAP

Critère → **Système ↓**	**Stockage**	**Requêtes**
ROLAP	SGBD relationnel	SQL & MDX
MOLAP	Hypercube	Moteur multidimensionnel
HOLAP	SGBD relationnel	Moteur multidimensionnel

Approches de conception

Nous notons deux principales visions méthodologiques de conception du système décisionnel d'une entreprise : l'approche *top-down* et l'approche *bottom-up*.

Approche *top-down* Cette solution est prônée par W. H. Inmon (1994), qui propose de concevoir l'entrepôt de données dans un premier temps et les magasins de données ensuite. Cela implique donc d'avoir *a priori* une vision intégrale des dimensions et faits à inclure dans le futur système décisionnel. Cette approche a l'avantage d'offrir un système « propre », c'est-à-dire intégré et cohérent. Cependant, sa mise en œuvre est longue et fastidieuse.

Approche *bottom-up* Cette approche a été mise en exergue par Kimball (1996). Elle consiste à concevoir d'abord les magasins de données, pour les intégrer *a posteriori* en

FIGURE 2.6 – Approches de conception du système décisionnel

un entrepôt de données global. Cela permet une relative rapidité de mise en œuvre, dans le sens où les magasins de données sont exploitables dès le début du processus. Mais la difficulté est seulement différée. En effet, les dimensions et niveaux de granularité adoptés dans les magasins de données sont souvent hétérogènes, ce qui entraîne une complexité dans leur intégration et donc dans la conception de l'entrepôt de données.

Nous précisons qu'une approche hybride a aussi été proposée par KIMBALL et ROSS (2013), dans le but de retenir le meilleur de chacune des deux visions.

2.2.5 Mise en oeuvre d'une architecture décisionnelle

Processus d'intégration des données

Une fois les outils décisionnels conçus, les données y sont intégrées à partir des sources de données via le processus *Extract-Transform-Load* (ETL). Ce processus consiste à obtenir les données des sources de données (*Extract*), à les traiter (*Transform*) avant de les insérer dans l'entrepôt ou le magasin de données (*Load*). Ces trois étapes peuvent être détaillées comme suit (KIMBALL & ROSS, 2013).

Phase d'extraction Elle consiste à recueillir les données à partir des différentes sources des données. La mise en place de la phase d'extraction des données passe avant tout par un profilage des sources de données, c'est-à-dire une analyse de la structure et de la nature des données. Il faut ensuite mettre en place un système automatisé d'extraction des données, qui sont généralement extraites et transférées dans un ODS à travers des fichiers, ou alors via des flux de données (*streams*). Dans certains cas, les données extraites sont préalablement cryptées et compressées pour des besoins de sécurité et de rapidité du transfert des données.

Phase de transformation Elle consiste à nettoyer et à organiser les données de sorte à les conformer au système cible. Plus concrètement, cette étape permet de détecter et de corriger d'éventuelles incohérences dans les données (valeurs manquantes, valeurs erronées, violation de hiérarchies de dimensions, etc.) avant leur transfert dans un système OLAP. La phase de transformation sert aussi et surtout à reformater les données provenant de systèmes divers en une structure compatible avec le modèle multidimensionnel (séparation des dimensions, dénormalisation, etc.).

Phase de chargement Elle consiste à transférer les données dans l'entrepôt ou le magasin de données à l'aide d'un ensemble de micro-programmes. Il est nécessaire d'établir un système d'ordonnancement et d'annulation du transfert des données de sorte à toujours garantir la cohérences des données dans le système OLAP en cas de bogues.

Analyses

Une fois l'entrepôt de données et les magasins de données conçus, implémentés et alimentés, les données sont alors prêtes pour les analyses. Nous distinguons deux principaux types d'analyses : les analyses OLAP et les analyses de type apprentissage automatique (*machine learning*).

Analyses OLAP Ces analyses permettent aux utilisateurs métiers de visualiser l'activité représentée par un entrepôt ou un magasin de données. L'analyse OLAP consiste plus concrètement à naviguer à travers les données dans l'optique de trouver et comprendre des tendances sur l'activité. On parle d'analyse exploratoire ou encore d'analyse en ligne.

Définition 2.2.4. L'acronyme « OLAP » est polysémique. L'analyse OLAP désigne une méthode d'analyse des données. On parle aussi d'analyse en ligne. En revanche, l'expression « systèmes OLAP » désigne des éléments de l'architecture décisionnelle qui suivent un modèle multidimensionnel.

L'analyse OLAP se fait à travers un ensemble d'opérateurs qui peuvent être regroupés en trois catégories (Favre, 2007 ; Teste, 2000).

1. **Les opérateurs de restriction** servent à restreindre les données analyser. Il s'agit typiquement des opérateurs *slice* et *dice* qui correspondent à une restriction en algèbre relationnelle ou en SQL.
2. **Les opérateurs liés à la granularité** exploitent les hiérarchies de dimension pour représenter les données suivant un niveau de détail plus ou moins agrégé. Le forage vers le haut (*roll-up*) permet ainsi d'aller vers un niveau de granularité supérieur. Par exemple, on pourrait passer d'une visualisation par mois à une visualisation par année en faisant un *roll-up* sur une dimension temporelle. Le forage vers le bas (*drill-down*) permet l'opération inverse.
3. **Les opérateurs liés à la structure** permettent de changer l'angle de vue des données sans les restreindre, ni changer de niveau de granularité. La rotation (*rotate*) permet ainsi de faire varier les dimensions par rapport aux valeurs présentées. La permutation (*switch*) échange deux membres d'une même dimension, dans le but par exemple de comparer leurs valeurs respectives. Enfin, la division (*split*) permet de séparer la visualisation globale des données en plusieurs sous-visualisations en omettant une dimension. Chaque sous-visualisation correspond alors à un membre de la dimension omise.

Apprentissage automatique Ce type d'analyses est communément en marge de l'architecture décisionnelle. Il exploite des données issues de l'ODS, de l'entrepôt ou des magasins de données. Il s'agit d'un ensemble de méthodes statistiques et d'intelligence artificielle utilisées pour réaliser des analyses descriptives avancées, mais aussi prédictives. De telles techniques peuvent par exemple servir à prédire le chiffre d'affaire futur de l'entreprise à partir des données historiques. Contrairement à l'analyse OLAP, ce type

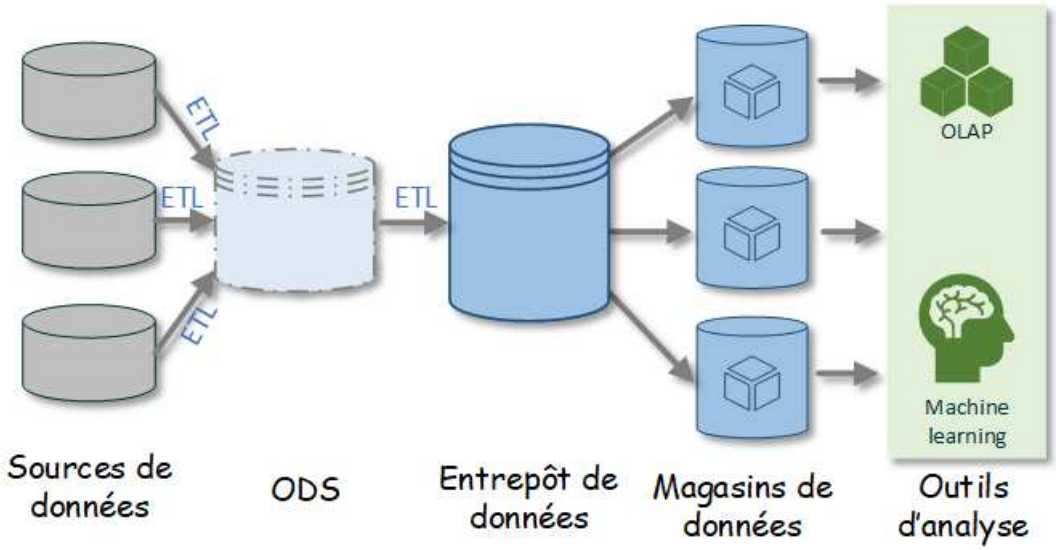


FIGURE 2.7 – Architecture décisionnelle détaillée

d'analyses nécessite une préparation supplémentaire et spécifique des données. C'est pourquoi ces analyses avancées sont réalisées de façon ponctuelle, par des experts du traitement de données (statisticiens, *data scientists*, *data analysts*, etc.).

La Figure 2.7 met à jour l'architecture décisionnelle en y ajoutant le processus ETL et en détaillant les outils d'analyse.

2.3 Limites et adaptations de l'entrepôt de données

2.3.1 Émergence des mégadonnées

Au début des années 2000, les systèmes classiques de stockage et d'analyse de données (OLTP et OLAP) ont été mis au défi par l'apparition et la disponibilité de nouvelles sources de données. Ces sources de données sont induites principalement par l'émergence des média sociaux (Wikipédia [1] en 2001, Facebook [2] en 2004, Twitter [3] en 2006, etc.), des objets connectés (*Internet of Things – IoT*), mais aussi par la croissance des systèmes d'information internes des entreprises, qui disposent désormais d'un volume important de courriels, de fichiers journalisés (*logs*), d'images, etc. (AJAH & NWEKE, 2019). On parle de mégadonnées ou plus communément de *big data*.

Définition 2.3.1. Les mégadonnées ou *big data* sont des données de tailles et formats si complexes qu'elles excèdent les capacités des systèmes informatiques traditionnels pour leur stockage et leur traitement (MILOSLAVSKAYA & TOLSTOY, 2016).

Les mégadonnées peuvent aussi être définies par des caractéristiques en V, dont les plus communément citées sont le volume, la variété et la vélocité (Figure 2.8). Plus

1. https://fr.wikipedia.org/
2. https://fr-fr.facebook.com/
3. https://twitter.com/

concrètement, ces caractéristiques peuvent être définies comme suit (Krishnan, 2013 ; Miloslavskaya & Tolstoy, 2016).

— Le **volume** représente la taille croissante des données à traiter et à stocker. La taille des données produites annuellement à travers le monde est ainsi passée de 2 zettaoctets (2 milliards de terraoctets) en 2010 à 47 zettaoctets en 2020, et devrait même atteindre 175 zettaoctets en 2025 (Gaudiaut, 2020).

— La **variété** met en exergue la forte hétérogénéité des données, notamment concernant le format. Ainsi, les mégadonnées incluent à la fois des données structurées provenant des systèmes OLTP, des données semi-structurées (pages web, *logs*, etc.) et des données non structurées (documents textuels, images, audios, vidéos, etc.).

— La **vélocité** décrit la rapidité de génération des données. C'est le cas des données de capteurs et de celles issues des médias sociaux. Ces données arrivent très souvent en flux continu et nécessitent donc un traitement en temps réel.

En plus de ces trois caractéristiques en V, d'autres sont également associées aux mégadonnées sous la forme de problématiques. Il s'agit notamment des problèmes posés par la *véracité*, la *valeur* ou encore la *visibilité* (accessibilité) des données.

Cette énorme masse de données a offert de nouvelles possibilités d'analyses qui intéressent les organisations. L'analyse des mégadonnées peut ainsi servir à améliorer l'offre de santé (Rangarajan et al., 2018). Dans le domaine de l'éducation, ces analyses permettraient de mieux mesurer l'efficacité des enseignements et donc de les améliorer. Elles permettent également de détecter, voire d'anticiper la saturation du trafic dans les grandes villes et d'agir conséquemment (Ajah & Nweke, 2019).

En offrant de nouvelles opportunités d'analyses, les mégadonnées engendrent aussi de nouvelles problématiques étroitement liées à leurs caractéristiques intrinsèques (volume, variété, vélocité, etc.). Le système d'entreposage traditionnel des données s'est ainsi montré limité et des variantes d'entrepôts/magasins de données ont alors été proposées.

2.3.2 Adaptations de l'entreposage de données

Plusieurs adaptations ont été proposées au système traditionnel d'entreposage des données pour faire face aux nouveaux défis liés aux *big data*. En guise d'illustration, nous présentons dans cette section des variantes du système d'entreposage introduites pour remédier aux trois principales problématiques engendrées par les *big data*, à savoir le volume, la vélocité et la variété des données.

Problématique du volume de données

Au fil des années, le volume des données stockées dans les entrepôts de données a explosé. Cela est notamment dû au fait que l'historique des données y est conservé (W. H. Inmon, 2005). Une autre raison de cette explosion est la multiplication des sources de données avec les *big data*. Cet état de fait a alors engendré un problème de passage à l'échelle des entrepôts de données. En effet, les entrepôts de données étant communément implémentés via des SGBD relationnels, ils sont soumis aux contraintes de ces systèmes. Or, les SGBD relationnels se doivent de garantir des propriétés d'atomicité, de cohérence, d'isolation

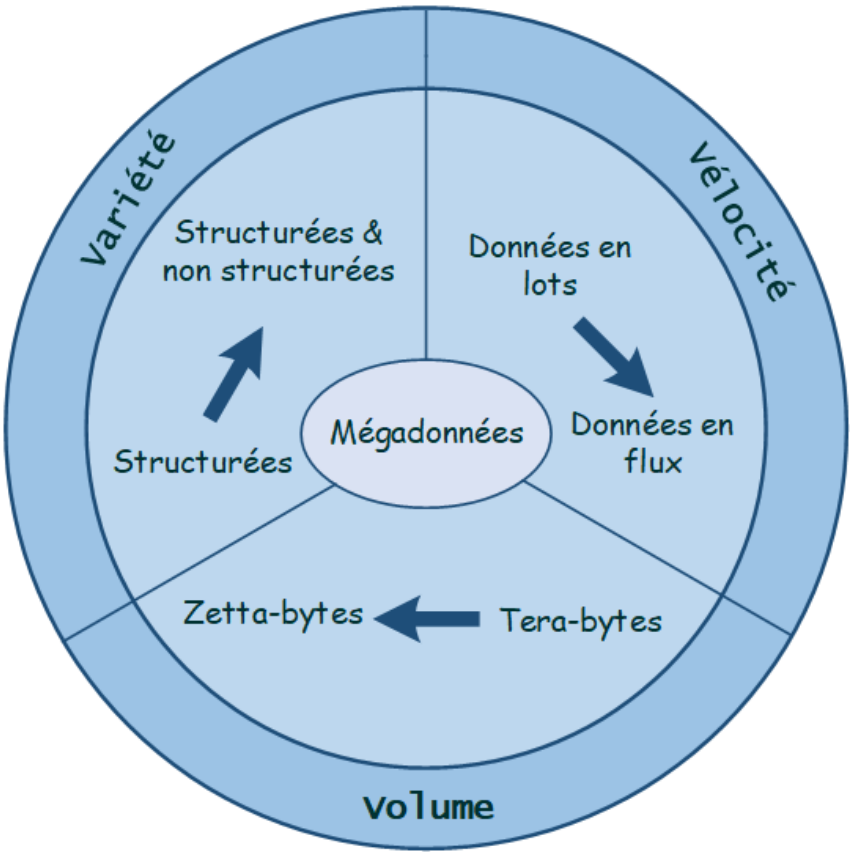


FIGURE 2.8 – Caractéristiques principales des mégadonnées (AJAH & NWEKE, 2019)

et de durabilité (ACID), lesquelles sont incompatibles avec un stockage et un traitement distribués, pourtant nécessaires pour passer à l'échelle (KRISHNAN, 2013).

Pour remédier aux limites du modèle relationnel, une nouvelle catégorie de SGBD a emergé dans les années 2000. Ces SGBD, appelés NoSQL (*Not Only SQL*), s'affranchissent des contraintes ACID et adoptent un système de stockage et de traitement distribués pour assurer le passage à l'échelle (MONIRUZZAMAN & HOSSAIN, 2013). L'innovation apportée par les SGBD non relationnels a marqué une nouvelle ère dans la conception des entrepôts de données. Le passage à l'échelle peut désormais être assuré grâce à des implémentations non relationnelles d'entrepôts de données. Cependant, cette solution nécessite une redéfinition du mode d'interrogation des données car les standards SQL et MDX ne sont pas nativement supportés dans de tels systèmes.
Nous proposons une description plus approfondie des SGBD non relationnels dans la Section 4.3.2.

Problématique de la vélocité des données

Les données sont traditionnellement intégrées dans un entrepôt de données via un processus ETL périodique (journalier, hebdomadaire, etc.). Cette intégration non-continue des données entraîne un décalage temporel entre les données des sources et celles qui sont disponibles dans l'entrepôt (BRUCKNER et al., 2002). Les besoins d'analyses de données en temps réel nécessitent alors des alternatives aux approches classiques d'entreposage

des données, d'autant que les données ne sont généralement pas disponible pour l'analyse pendant le processus ETL (Polyzotis et al., 2007).

Dès le début des années 2000, plusieurs travaux ont traité la problématique d'intégration des données en temps réel et plusieurs stratégies ont été proposées. Par exemple, Bruckner et al. (2002) ont proposé un système à base de conteneurs permettant de transformer en mémoire vive les données extraites, pour les intégrer directement dans l'entrepôt (sans passer par un stockage intermédiaire, comme c'est communément le cas). Polyzotis et al. (2007) ont eux abordé la problématique sous-jacente d'assimilation en temps réel des données nouvelles aux anciennes dans un entrepôt de données. Leur approche permet ainsi de garantir la cohérence des données de l'entrepôt, sans pour autant nécessiter un arrêt des services lors de l'intégration des données.

La problématique d'entreposage en temps réel des données est restée une question de recherche active jusqu'à la dernière décennie et l'émergence des lacs de données (Rehman et al., 2012 ; Zhou et al., 2011). D'ailleurs, la plupart des approches proposées pour l'entreposage en temps réel des données ne remédient qu'à une partie du problème. En effet, l'entreposage en temps réel nécessite non seulement une intégration continue des données, mais aussi un système de décision automatisé ainsi qu'un accès ininterrompu aux données de l'entrepôt (Bruckner et al., 2002). Or, seule la dimension d'intégration continue est traitée, le plus souvent. De nouveaux outils et approches sont donc encore nécessaires pour répondre au besoin d'entreposage en temps réel.

Problématique de variété des données

Avec les *big data*, les sources de données évoluent continuellement. On assiste ainsi à l'apparition de nouvelles sources de données et au changement de sources de données préexistantes. Ces évolutions induisent donc une plus grande variété des données à analyser dans les systèmes décisionnels. Cette variété des données pose entre autres deux défis majeurs aux entrepôts de données traditionnels.

Entrepôts de données agiles Les évolutions dans les sources de données préexistantes se traduisent souvent par des changements dans le schema des données à intégrer dans l'entrepôt. Or, les approches traditionnelles d'entreposage des données sont peu agiles. Des modèles alternatifs de modélisation des entrepôts de données ont donc été proposés, parmi lesquels les approches de modélisation ensemblistes. Ces modèles adoptent un principe de décomposition unifiée. Cela consiste à séparer les attributs qui changent de ceux qui ne changent pas, au lieu de les encapsuler dans les mêmes entités (dimension, fait) comme dans l'approche classique. Ces différents éléments sont ensuite regroupés en « ensembles », d'où l'appellation « modélisation ensembliste » (Hultgren, 2016 ; Rönnbäck & Hultgren, 2013). Les deux approches de modélisation ensembliste les plus connues sont les *data vaults* et l'*anchor modeling*.

La modélisation en *data vault* a été inventée dans les années 1990, mais elle n'a été véritablement vulgarisée que dans les années 2010 par Linstedt (2011). Elle peut être vue comme une combinaison de la 3FN avec le modèle en étoile (Nogueira et al., 2018). La modélisation en *data vault* se base sur trois concepts.

1. Le ***hub*** représente le coeur d'un concept métier de base (par exemple : client, vendeur, produit, etc.). Il contient uniquement la clé naturelle (*business key*) et ne contient pas d'attribut descriptif, ni de clé étrangère.
2. Le **satellite** contient des informations descriptives associées à un *hub* ou à un *lien*.
3. Le **lien** (*link*) permet d'établir une correspondance entre des instances de deux (ou plusieurs) *hubs*. Il contient à minima deux clés étrangères qui représentent les deux *hubs* à associer, et peut lui-même être associé à un satellite.

En séparant les concepts de base de leur contexte, le modèle en *data vault* assure l'évolutivité ainsi que l'historisation des informations contextuelles.

> *Exemple :* Pour illustrer le fonctionnement de la modélisation en *data vault*, nous proposons le cas d'une société de livraison de médicaments, en nous inspirant de l'exemple de Krneta et al. (2014). Nous considérons que cette entreprise dispose d'un entrepôt simpliste, constitué d'un fait « livraison » et de trois dimensions à savoir « produit », « fournisseur » et « pharmacie ». La Figure 2.9-A montre une représentation logique relationnelle du modèle en flocons de neige correspondant à cet entrepôt (nous considérons une liaison hiérarchique entre la dimension « produit » et la dimension « fournisseur »).
>
> En traduisant ce modèle en *data vault*, on obtient trois *hubs* (« produit », « fournisseur » et « pharmacie »), associés chacun à un satellite, comme le montre la figure 2.9-B. Le fait « livraison » se traduit quant à lui par un lien éponyme, également associé à un satellite.

L'*anchor modeling* a été introduite par Regardt et al. (2009) et Rönnbäck et al. (2010). Elle se base sur quatre concepts : l'ancre (*anchor*), le nœud (*knot*), le lien (*tie*) et l'attribut.

1. L'**ancre** représente une entité (client, magasin, etc.). Chaque instance de ce concept est représentée par une clé. Contrairement aux autres concepts, l'ancre n'est pas historisée.
2. Les **attributs** représentent des valeurs historisées ou non. Chaque attribut est associé à une ancre et contient donc l'identifiant de cette ancre, la valeur stockée et éventuellement un horodatage.
3. Les **liens** servent à établir une relation entre deux ou plusieurs ancres. Le lien peut être associé à des attributs.
4. Le **nœud** permet d'associer une ancre à un attribut. Ce concept est en réalité facultatif et sert surtout à simplifier le modèle

Ces concepts permettent à l'*anchor modeling* d'être encore plus agile que la modélisation *data vault*. Elle se conforme ainsi à la 6FN (contrairement à la 3FN pour le *data vault*). Mais cela induit une plus grande complexité de conception et d'implémentation (Nogueira et al., 2018).

> *Exemple :* La Figure 2.10 présente une conversion en *anchor modeling* des modèles en flocon de neige et *data vault* de la Figure 2.9. Nous avons évité l'utilisation de nœuds, par souci de clarté.

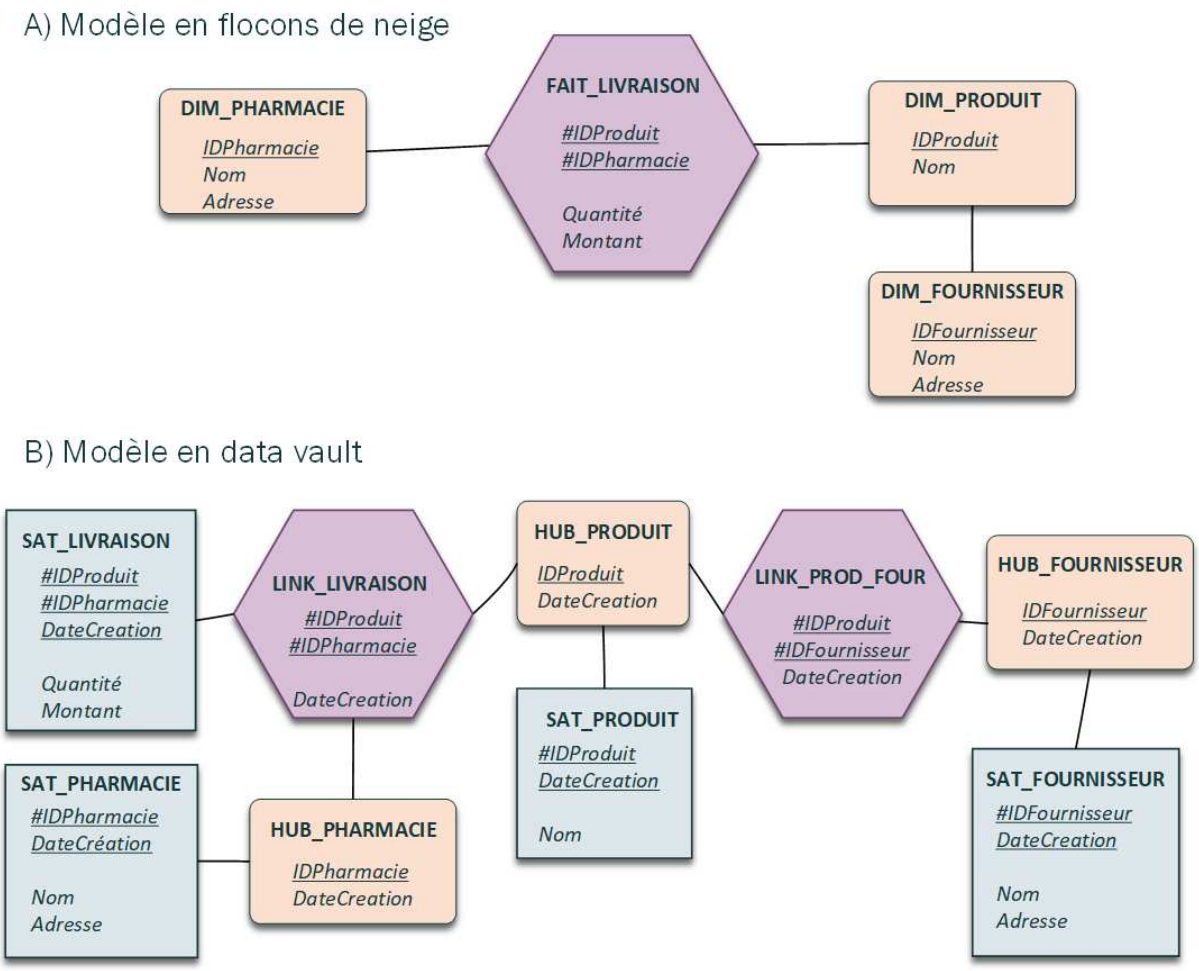


FIGURE 2.9 – Exemple de modélisation en *data vault*

Nous constatons en effet dans l'exemple de la Figure 2.10 une complexification du modèle obtenu par rapport au modèle en *data vault* de la Figure 2.9-B, et encore plus par rapport au modèle en flocons de neige de la Figure 2.9-A. Pour notre exemple, l'implémentation relationnelle de ces trois modèles conduit ainsi à 4 tables pour le modèle en flocons de neige, 9 tables pour le modèle en *data vault* et 12 tables pour le modèle en ancre.

Entrepôts de données textuelles L'apparition de nouvelles sources de données a aussi conduit à des besoins d'analyses plus étendus. En effet, les entreprises souhaitent réaliser des analyses systématisées sur les grands volumes de données à leur disposition (*big data*). Or, la majorité de ces données est non structurée, et donc inadaptée au processus classique d'entreposage qui est traditionnellement réservé aux données structurées (MILOSLAVSKAYA & TOLSTOY, 2016). C'est pourquoi, des variantes d'entrepôts de données ont été proposées pour prendre en compte les données semi-structurées et non structurées, notamment textuelles.

Dans un entrepôt de données textuelles, les dimensions sont soit des métadonnées des documents (titre, date de création, auteur, etc.), soit des thématiques (W. H. INMON, 2005). Plusieurs méthodes ont ensuite été proposées pour agréger les données textuelles

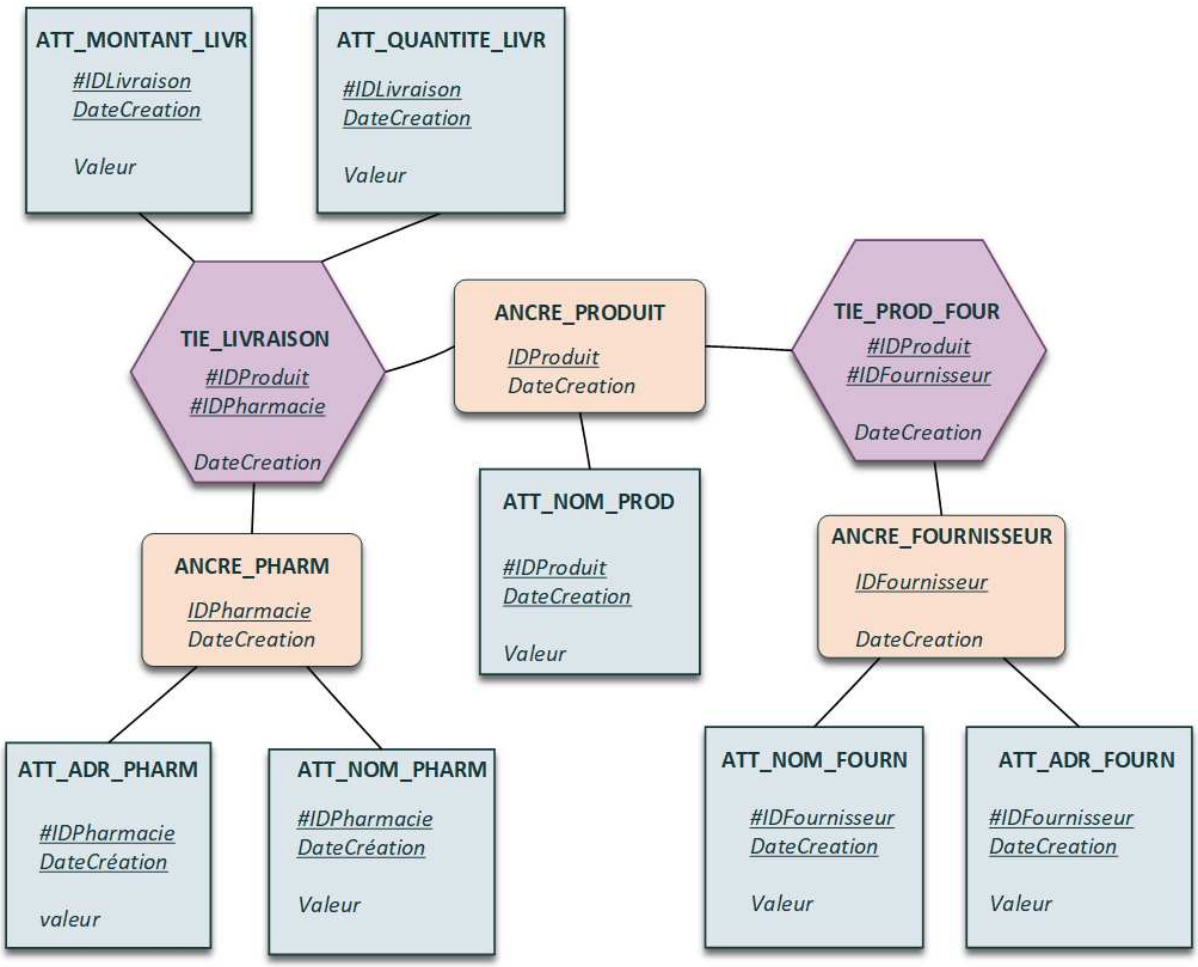


FIGURE 2.10 – Exemple d'*anchor modeling*

lors des analyses. Ces approches peuvent être regroupées en trois catégories (AZABOU et al., 2016).

1. Les **méthodes basées sur la fouille de texte (*text mining*)** utilisent des techniques d'intelligence artificielle pour résumer les documents à travers des thématiques automatiquement détectées (LIU et al., 2009 ; MESSAOUD et al., 2004).
2. Les **méthodes basées sur la recherche d'information** conçoivent le document comme un simple ensemble de termes et agrègent donc les documents sur la base d'un décompte de ces mots clés. Par exemple, RAVAT et al. (2008) proposent de représenter un ensemble de documents agrégés par les mots clés les plus fréquents qui y apparaissent.
3. Les **méthodes utilisant des bases de connaissances** agrègent les documents en se servant de dictionnaires, thesauri ou ontologies. RAVAT et al. (2007) introduisent ainsi l'opérateur d'agrégation AVG_KW qui exploite une ontologie pour résumer des documents (représentés par un ensemble de termes) en un nombre réduit de mots clés représentatifs. De façon analogue, OUKID et al. (2016) proposent l'opérateur d'agrégation *ORank* pour classer des documents sur la base de préférences fournies par l'utilisateur.

Ces méthodes permettent ainsi à des utilisateurs métiers d'explorer des corpus de documents de façon analogue à l'analyse OLAP. On parle de *Text-OLAP*.

Cependant, les entrepôts de données textuelles ne satisfont pas entièrement l'évolution des besoins d'analyse. En effet, les entrepôts de données textuelles sont conçus sur des infrastructures et à l'aide de technologies souvent séparées de celles utilisées par les entrepôts de données traditionnels (W. H. Inmon, 2005). Cela implique une séparation des données et donc une limitation des possibilités d'analyses croisées entre données structurées et données non structurées. De plus, les entrepôts de données textuelles contiennent uniquement des documents transformés et font ainsi abstraction des données brutes. Cela engendre une perte d'informations et limite le spectre des analyses possibles.

2.4 Lacs de données

2.4.1 De la technologie Hadoop aux lacs de données

Si des variantes d'entrepôts de données permettent de répondre à certains défis engendrés par les *big data*, ces solutions restent tout de même limitées. En effet, chaque variante d'entrepôt de données ne remédie qu'à une partie de la problématique globale. Par exemple, la problématique liée à la variété des données n'est que partiellement résolue par les entrepôts de données textuelles, puisque les fichiers images, audios et vidéos en sont exclus. De plus, ces solutions au cas par cas induisent une répétition de la tâche proportionnelle à la multiplicité des cas. Enfin, elles impliquent surtout une isolation des données en silos qui limite les analyses croisées entre les données des entreprises (Krishnan, 2013).

Une solution plus adaptée pour le stockage et l'exploitation des *big data* a alors été proposée à travers la technologie *Apache Hadoop*[4]. Cette technologie est née au milieu des années 2000. Elle intègre à la fois un système de stockage distribué des données appelé *Hadoop Distributed File System* (HDFS) et une plateforme de traitement parallèle des données intitulé *Map-Reduce* (Tiao, 2018). À travers ces deux composantes, Hadoop permet d'étendre les capacités de stockage et de traitement des données au sein des entreprises. Il prend ainsi en charge tous types de données et offre un stockage moins coûteux (Kimball & Ross, 2013).

La technologie Hadoop a été massivement adoptée au sein des entreprises et un nouveau paradigme de stockage et d'exploitation des données est apparu avec elle. En effet, les nouvelles capacités et le faible coût de stockage induits par la technologie Hadoop ont conduit à un stockage de plus en plus systématisé et sans limite de données de tous types au sein des entreprises (Krishnan, 2013). Contrairement à l'approche classique d'entreposage, les données sont stockées sous leur forme brute, indépendamment de la façon dont elles seront analysées. D'ailleurs, les données sont stockées pour un usage futur non prédéterminé (Stein & Morrison, 2014). Ce nouveau paradigme a été présenté comme le concept de lac de données.

4. https://hadoop.apache.org/

2.4.2 Définitions

Au cours de la dernière décennie, le concept de lac de données a été largement adopté, aussi bien en milieu académique que dans l'industrie. Cependant, certaines ambiguïtés demeurent quant à la définition exacte d'un lac de données. Une enquête conduite par Grosser et al. (2016) révèle en effet que 35 % des participants conçoivent le concept de lac de données comme un simple concept marketing. Pour désambiguïser le concept de lac de données, nous présentons ce que sont les lacs de données, d'une part, et ce qu'ils ne sont pas, d'autre part.

Ce que les lacs de données sont

Définition originelle Le concept de lac de données a été introduit par Dixon (2010) en guise de solution aux multiples inconvénients des magasins de données. En effet, la nature prétransformée et agrégée des données situées dans les magasins de données (et dans l'entrepôt) limite les possibilités d'analyses à des questions prédéfinies. Pour y remédier, James Dixon propose alors le concept de lac de données à travers une citation devenue célèbre.

Définition 2.4.1. Si vous considérez un magasin de données comme un stock d'eau embouteillée, nettoyée, emballée et structurée pour une consommation facile, le lac de données est une grande étendue d'eau dans un état plus naturel. Le contenu du lac de données afflue de sources pour remplir le lac et divers utilisateurs peuvent venir l'examiner, y plonger ou prendre des échantillons.

En d'autres termes, un lac de données désigne un vaste système de stockage de données brutes et hétérogènes, alimenté par de multiples sources de données et qui permet à divers utilisateurs d'explorer, d'extraire et d'analyser les données.

Dans la littérature, d'autres terminologies ont été utilisées pour désigner des systèmes analogues et donc assimilables au lac de données. On peut ainsi citer le concept de « réservoir de données » (Chessell et al., 2014), ou encore celui de « centre de données » (*data hub*) (Laskowski, 2016).

Définition consensuelle La grande majorité des définitions du concept de lacs de données se base sur deux caractéristiques essentielles : la variété des données et l'approche *schema-on-read* (Khine & Wang, 2017 ; Laskowski, 2016 ; Maccioni & Torlone, 2017 ; Mathis, 2017). La variété des données traduit le fait que les lacs de données supportent des données de tous types. La caractéristique *schema-on-read*, encore appelée *late binding*, désigne quant à elle le fait que les données sont intégrées sous leur forme brute, sans transformation *a priori*. Cette approche est à l'opposé de l'approche *schema-on-write* ou *early binding* utilisée dans les entrepôts de données. De là, le concept de lac de données peut être défini comme suit.

Définition 2.4.2. Un lac de données est un dépôt central où des données de tous formats sont stockées sans schéma prédéfini, en vue d'analyses futures.

Définition de Madera et Laurent La définition « variété/*schema-on-read* » est certes consensuelle, mais elle ne détaille pas suffisamment les caractéristiques d'un lac de données. Madera et Laurent (2016) proposent une définition plus complète.

Définition 2.4.3. Un lac de données est une vue logique de toutes les sources de données et de tous les ensembles de données dans leur format brut, accessible aux spécialistes des données ou aux statisticiens pour l'extraction de connaissances.

Cette définition est complétée par un ensemble de caractéristiques qu'un lac de données devrait inclure.

1. La qualité des données est assurée par un ensemble de métadonnées.
2. Le lac est contrôlé par des outils et une politique de gouvernance des données.
3. L'utilisation du lac est limitée aux statisticiens et aux scientifiques des données.
4. Le lac intègre des données de tous types et de tous formats.
5. Le lac de données a une organisation logique et physique.

Notre définition Certains points de la définition de Madera et Laurent (2016) nous paraissent discutables. En effet, l'utilisation du lac y est réservée aux spécialistes des données, excluant les utilisateurs métiers pour des raisons de sécurité. Pourtant, il est tout à fait envisageable de proposer un accès contrôlé et adapté à ce type d'utilisateurs par le biais d'une interface graphique d'exploration ou d'analyses.

De plus, nous ne partageons pas la vision du lac de données comme une vue logique sur les sources de données, puisque certaines sources de données peuvent être externes à une organisation et donc au lac de données. D'ailleurs, comme Dixon (2010) indique explicitement que les données du lac proviennent de sources de données, inclure des sources de données dans le lac nous paraît contraire à l'esprit des lacs de données.

Enfin, bien qu'étant assez complète, la définition de Madera et Laurent (2016) omet une propriété essentielle des lacs de données : le passage à l'échelle (*scalability*) (Khine & Wang, 2017 ; Miloslavskaya & Tolstoy, 2016). Le lac de données étant destiné au stockage et au traitement des *big data*, il est en effet essentiel de traiter cette problématique.

Pour toutes ces raisons, nous proposons une évolution de la définition de Madera et Laurent (2016) qui se veut plus conforme à notre vision des lacs de données :

Définition 2.4.4. Un lac de données est un système évolutif de stockage et d'analyse de données de tous types, conservées dans leur format natif et utilisées *principalement* par des spécialistes des données (statisticiens, *data scientists* ou *data analysts*) pour l'extraction de connaissances.

Comme Madera et Laurent (2016), nous proposons également un ensemble de six préceptes auxquels un lac de données doit se conformer.

1. Un lac de données inclut un catalogue de métadonnées qui assure la qualité des données.
2. Un lac de données comprend des politiques et des outils de gouvernance des donnée.

3. Un lac de données est accessible à divers types d'utilisateurs.
4. Un lac de données intègre tous types de données.
5. Un lac de données a une organisation logique et physique.
6. Un lac de données doit pouvoir passer à l'échelle.

Ce que les lacs de données ne sont pas

Lacs de données et Apache Hadoop Bien que Dixon (2010) ait formalisé le lac de données comme un concept générique, indépendant des technologies sous-jacentes, une partie de la littérature l'a tout de même systématiquement associé à la technologie Hadoop (Fang, 2015 ; Ganore, 2015 ; O'Leary, 2014). Ce faisant, le lac de données a alors souvent été perçu comme un équivalent de Hadoop ou plus généralement des technologies libres (*open source*). La définition de Fang (2015) entre dans ce cadre.

Définition 2.4.5. Le concept de lac de données désigne une méthodologie permettant d'utiliser des technologies gratuites ou peu coûteuses, typiquement Hadoop, pour stocker, traiter et explorer des données brutes au sein d'une entreprise.

Cette vision est de plus en plus minoritaire dans la littérature, puisque le concept de lac de données est désormais également associé à des solutions *cloud* propriétaires telles que Microsoft Azure [5] ou IBM (Madera & Laurent, 2016 ; Sirosh, 2016) et à divers systèmes de gestion de données tels que les SGBD NoSQL et les multistores (Section 4.3.2).

Lacs et marécages de données En l'absence d'un schéma prédéfini dans un lac, l'accès aux données et les analyses dépendent d'un système de métadonnées qui se doit d'être efficace. Or, il arrive que le système de métadonnées soit insuffisant ou même inexistant. À ce moment, le lac de données devient un *data swamp*, c'est-à-dire, un « marécage de données » (Chessell et al., 2014 ; Khine & Wang, 2017). Un tel système est alimenté par des données brutes, organisées de façon si anarchique que l'on ne peut en extraire de la valeur. C'est pourquoi B. Inmon (2016) et Suriarachchi et Plale (2016) s'y réfèrent à travers l'expression « dépotoir de données ».

2.4.3 Lacs et entrepôts de données

Dixon (2010) a défini le concept de lac de données en opposition aux systèmes OLAP traditionnels (entrepôts et magasins de données). Dans cette section, nous poussons plus loin la comparaison entre lacs et entrepôts de données, en détaillant les dissemblances entre ces concepts. Ces différences peuvent être regroupées du point de vue des entrées (ingestion des données) et des sorties (analyses).

Ingestion des données

Une première différence fondamentale entre les entrepôts et les lacs de données est énoncée dans la définition de Dixon (2010) : les entrepôts de données traitent des « données

5. https://azure.microsoft.com/fr-fr/solutions/data-lake/

nettoyées », tandis que les données d'un lac sont dans un « état plus naturel », c'est-à-dire qu'elles restent dans un format brut.

De ce fait, la façon dont les données sont préparées pour l'analyse diffère. Dans un entrepôt de données, cette préparation suit généralement un processus continu d'extraction, de transformation et de chargement (ETL). Dans un lac de données, les données ne sont pas transformées *a priori* (O'Leary, 2014). Le processus d'ingestion des données y suit plutôt une approche *Extract-Load-Transform* (ELT) (Avinoam, 2018), où la phase de transformation des données est différée.

L'approche ELT est parfois utilisée dans les entrepôts de données, mais l'ensemble du processus reste continu et seules les données transformées sont conservées. Le processus typique d'entreposage de données peut ainsi entraîner des pertes d'informations, alors que les lacs de données garantissent une fidélité aux données (Stein & Morrison, 2014), notamment parce que les données originales sont normalement conservées.

O'Leary (2014) souligne une autre différence concernant la diversité des sources de données. L'entrepôt de données étant construit pour répondre à des questions prédéfinies, les sources de données y sont limitées à celles qui répondent aux besoins d'analyses. Tel n'est pas le cas du lac de données où un plus grand nombre de sources de données est exploité.

Enfin, en l'absence de prétraitements spécifiques, les lacs de données peuvent stocker n'importe quel type de données, c'est-à-dire des données structurées, semi-structurées et non structurées, éventuellement toutes ensemble. En revanche, les systèmes OLAP classiques nécessitent des données structurées en entrée (nous excluons volontairement les propositions d'adaptation d'entrepôts de données qui ne sont pas couramment utilisées). Ainsi, même si des données semi-structurées sont introduites dans un entrepôt de données, elles sont transformées en un format structuré avant d'être intégrées (O'Leary, 2014).

Analyse des données

La principale différence entre les entrepôts et les lacs de données réside vraisemblablement dans la gestion du schéma des données. Les entrepôts de données suivent une approche *schema-on-write*, où le schéma est fixé *a priori* par rapport aux exigences d'analyses. Les lacs de données suivent quant à eux une approche *schema-on-read*, dans laquelle les exigences sur le schéma des données ne sont définies qu'au moment de leur interrogation (Ansari et al., 2018 ; Khine & Wang, 2017 ; Stein & Morrison, 2014).

Comme les entrepôts de données ont un schéma fixe et prédéfini, ils s'adaptent plus difficilement aux évolutions de schémas et supportent donc difficilement de nouveaux besoins d'analyses. De plus, les opérations de mise à jour des données (dans les dimensions, essentiellement) leur posent de sérieux problèmes de gestion. En revanche, les lacs de données offrent aux spécialistes des données la possibilité de configurer facilement des requêtes et des applications à la volée (Khine & Wang, 2017). C'est pourquoi ils sont considérés comme étant plus flexibles et plus agiles que les entrepôts de données.

Une autre différence liée au schéma concerne l'accès aux données. Les entrepôts de données sont conçus pour des analyses récurrentes et industrialisées, notamment via les langages standards SQL et MDX. En revanche, l'extraction et l'analyse des données dans les lacs

de données se font de façon plus ponctuelle via des scripts ou des programmes spécifiques (Fang, 2015).

La différence de mode d'accès aux données induit une autre différence concernant les profils d'utilisateurs. En effet, l'exigence d'un traitement spécifique des données pour chaque analyse ou opération d'extraction de données implique un besoin de compétences en traitement des données pour l'exploitation des lacs. Par conséquent, les utilisateurs des lacs de données sont le plus souvent des analystes et des scientifiques des données (Khine & Wang, 2017 ; Madera & Laurent, 2016). En revanche, les entrepôts de données peuvent être exploités par des utilisateurs n'ayant pas de compétences particulières en matière de gestion des données, notamment les utilisateurs métiers.

Enfin, la sécurité des données et le contrôle d'accès diffèrent sensiblement entre les entrepôts et les lacs de données. À cet égard, les entrepôts de données sont clairement en avance. La protection des données sensibles dans les entrepôts de données a en effet été améliorée depuis des décennies et peut désormais être considérée comme mature. Ce n'est pas le cas pour les lacs de données, où la sécurité des données reste un domaine de recherche ouvert (Khine & Wang, 2017).

Le Tableau 2.3 présente une synthèse des principales différences entre les entrepôts et les lacs de données.

Table 2.3 – Principales différences entre entrepôts et lacs de données

Système → **Critère ↓**	**Entrepôt de données**	**Lac de données**
État des données	Données nettoyées	Données brutes
Processus d'ingestion	*Extract-Transform-Load*	*Extract-Load-Transform*
Perte de données	Oui, lors du processus ETL	Fidélité aux données
Sources de données	Limitées	Illimitées
Format des données	Données structurées	Tous formats
Définition du schema	*Schema-on-write*	*Schema-on-read*
Flexibilité des analyses	Peu flexibles	Très flexibles
Types d'analyses	Prédéfinies, industrialisées via des langages de requêtes	À la volée, *ad-hoc*, via des scripts
Utilisateurs	Utilisateurs métiers	Spécialistes des données
Securisation des accès	Mature	En maturation

2.4.4 Opportunités et défis associés aux lacs de données

Le concept de lac de données offre plusieurs avantages qui répondent aux limites des entrepôts de données. Cependant, les lacs de données induisent aussi de nouveaux défis et

préoccupations (appelés les « alligators dans le lac ») qui nécessitent une certaine maturation (O'Leary, 2014). Dans ce qui suit, nous énumérons dix avantages et dix défis apportés par le concept de lac de données.

Opportunités

1. **Fidélité aux données :** Contrairement aux systèmes OLAP traditionnels, les données originales sont préservées dans un lac afin d'éviter toute perte de données qui pourrait provenir des opérations de prétraitement et de transformation des données (Ganore, 2015 ; Stein & Morrison, 2014).
2. **Compatibilité aux données non structurées :** L'un des principaux avantages des lacs de données est qu'ils permettent d'exploiter et d'analyser des données non structurées (Laskowski, 2016 ; Stein & Morrison, 2014). Cela constitue un avantage significatif pour l'exploitation des *big data*, dont la grande majorité est non structurée.
3. **Ingestion de données en temps réel :** Les données sont ingérées dans les lacs de données sans aucune transformation, ce qui limite le décalage temporel entre l'extraction des données des sources et leur ingestion dans le lac (Ganore, 2015 ; Laskowski, 2016).
4. **Stockage peu coûteux :** Les lacs de données sont dix à cent fois moins chers à déployer que les systèmes décisionnels traditionnels. Cela peut être attribué à l'utilisation de technologies libres comme Hadoop (Khine & Wang, 2017 ; Stein & Morrison, 2014). Une autre raison réside dans les avantages des services *cloud* souvent utilisés pour construire les lacs de données. En effet, les coûts de stockage y sont réduits par rapport à une infrastructure locale, car on n'y paie que pour les ressources utilisées.
5. **Passage à l'échelle :** Les lacs de données sont généralement mis en œuvre en utilisant des technologies distribuées, comme Hadoop, qui offrent de précieuses capacités de passage à l'échelle (Fang, 2015 ; Miloslavskaya & Tolstoy, 2016)
6. **Tolérance aux pannes** : La plupart des technologies utilisées dans les lacs de données utilisent des mécanismes de réplication. Cela procure une grande résilience face aux éventuelles défaillances matérielles et logicielles (John & Misra, 2017).
7. **Flexibilité et agilité :** Grâce à l'approche *schema-on-read*, les lacs de données peuvent se conformer à tout type et format de données. Par conséquent, ils supportent un plus large éventail d'analyses par rapport aux systèmes OLAP traditionnels.
8. **Analyses à la volée :** Les lacs de données sont souvent considérés comme des bacs à sable où les spécialistes des données peuvent « jouer », c'est-à-dire accéder aux données et les manipuler pour en extraire de la valeur.
9. **Découverte de liens entre des données hétérogènes :** L'intégration de données structurées, semi-structurées et non structurées dans le lac permet de détecter des corrélations entre des données de différents types (Ganore, 2015).
10. **Analyses croisées avec des données externes :** Les lacs de données permettent d'intégrer facilement des données provenant de sources externes, par exemple du

Web ou des médias sociaux. Ces données externes peuvent alors être associées aux données internes pour générer de nouvelles connaissances grâce à des analyses croisées (Laskowski, 2016).

Défis

1. **Confusions :** Le concept de lac de données reste ambigü pour de nombreux utilisateurs potentiels. Il est en effet souvent considéré comme un synonyme ou un label marketing étroitement lié à la technologie Hadoop (Alrehamy & Walker, 2015 ; Grosser et al., 2016).
2. **Absence de standards :** Bien que le concept de lac de données ait été introduit en 2010, il n'a véritablement émergé qu'au milieu des années 2010. Par conséquent, les approches d'implémentation sont toujours en cours de maturation et il n'existe pas encore de standard méthodologique ni technique. Tout cela entretient les confusions autour du concept de lac de données.
3. **Incohérence de données :** Il existe un risque élevé d'incohérence de données dans les lacs, en raison de l'intégration de données provenant de sources multiples et disparates sans aucune transformation (O'Leary, 2014).
4. **Nécessité d'un système de métadonnées :** Un lac de données nécessite un système de métadonnées efficace. Cependant, le problème réside dans le « comment ». En effet, l'utilisation de méthodes ou de technologies inappropriées pour construire le système de métadonnées peut facilement transformer le lac de données en un marécage de données inexploitable (Alrehamy & Walker, 2015).
5. **Besoin d'un service d'accès aux données :** En raison de l'absence de schéma explicite, des interfaces de programmation (*Application Programming Interfaces* ou API) sont essentielles pour permettre l'extraction de connaissances dans un lac de données. En d'autres termes, un service d'accès aux données est indispensable pour réussir la construction d'un lac (Alrehamy & Walker, 2015 ; B. Inmon, 2016).
6. **Besoin d'expertise :** Les utilisateurs d'un lac de données sont généralement des *data scientists*, ce qui contraste avec les systèmes OLAP traditionnels, où les utilisateurs métiers sont capables de faire fonctionner le système. Par conséquent, un lac de données induit un besoin accru de profils spécifiques et est donc plus coûteux à exploiter.
7. **Approches d'intégration et de transformation des données :** Bien que retardées, ces tâches sont toujours présentes dans les lacs de données. Elles y sont d'ailleurs compliquées par le volume, la variété, la vélocité et le manque de véracité des données. De plus, lors de la transformation de ces données, des fonctions définies par l'utilisateur (*User-Defined Functions* ou UDF) doivent très souvent être utilisées (tâches MapReduce) alors que ces UDF sont beaucoup plus difficiles à optimiser que les requêtes classiques (Stefanowski et al., 2017).
8. **Exploitation de données non structurées :** Les données non structurées, bien qu'unanimement reconnues comme étant omniprésentes et sources d'informations cruciales, sont très peu abordées spécifiquement dans la littérature relative aux lacs de données. Le stockage d'index et la fouille de texte sont généralement mentionnés, mais il n'y a pas de réflexion approfondie sur les solutions globales d'interrogation

ou d'analyse. De plus, l'exploitation d'autres types de données non structurées que le texte, par exemple les images, les sons et les vidéos, est encore très marginale.

9. **Gouvernance des données :** Là encore, bien que tous les acteurs du domaine des lacs de données soulignent l'importance de la gouvernance des données pour éviter qu'un lac de données ne se transforme en marécage de données, la qualité des données, la sécurité, la gestion du cycle de vie et le lignage des métadonnées sont considérés comme des risques plutôt que comme des questions à traiter *a priori* dans les lacs de données (Madera & Laurent, 2016). Les principes de gouvernance des données sont en effet actuellement rarement transformés en solutions réelles.
10. **Sécurisation des données :** Elle est actuellement abordée d'un point de vue technique dans les lacs de données, c'est-à-dire par le contrôle des accès et des privilèges, l'isolation du réseau, par exemple avec les outils Docker (Cha et al., 2018), le cryptage des données et les moteurs de recherche sécurisés (Maroto, 2018). Cependant, au-delà de ces aspects techniques, les lacs de données permettent de stocker et d'analyser de manière croisée de grands volumes de données diverses ; ce qui induit potentiellement de graves violations de la confidentialité des données (Joss, 2016). Ces questions font encore l'objet de recherches à l'heure actuelle.

2.5 Conclusion

Depuis l'apparition des SGBD relationnels, les systèmes dédiés au stockage ou à l'analyse des données évoluent continuellement, sous l'influence à la fois des besoins de stockage et d'analyse, mais aussi des technologies existantes. Ainsi, à la suite des SGBD relationnels qui répondaient avant tout à un besoin de stockage, ont émergé dans les années 1990 les systèmes multidimensionnels (entrepôts et magasin de données) répondant quant à eux à des besoins d'analyse.

Dans les années 2000, les besoins d'extensions des analyses, en conjonction avec l'apparition des SGBD NoSQL (et des approches de modélisation ensemblistes) ont conduit à la proposition de variantes d'entrepôts et de magasins de données (entrepôts de données textuelles, entrepôts temps réel, etc.). Enfin, l'apparition de la technologie Apache Hadoop en 2006 a favorisé l'émergence d'un nouveau paradigme qui répond encore mieux à l'extension continue des besoins d'analyse. Ainsi est né le concept de lac de données en 2010 (Figure 2.11).

Cependant, les lacs de données ne viennent pas remplacer, mais compléter les systèmes décisionnels préexistants. D'ailleurs, les différents systèmes proposés au fil des années ont toujours été mus par le besoin d'avoir un outil adapté à chaque besoin. C'est justement ce principe qui a prévalu à l'introduction des systèmes mutidimensionnels. Codd et al. (1993) affirme en effet que « tenter de forcer une technologie ou un outil à satisfaire un besoin particulier pour lequel un autre outil est plus efficace et efficient, c'est comme essayer d'enfoncer une vis dans un mur avec un marteau au lieu d'un tournevis. » Les systèmes multidimensionnels répondaient ainsi à un nouveau besoin pour lequel les SGBD relationnels n'avaient pas été conçus. C'est exactement le même principe qui a prévalu à l'apparition des lacs de données.

Si le concept de lac de données a atteint une certaine maturité, de multiples défis et

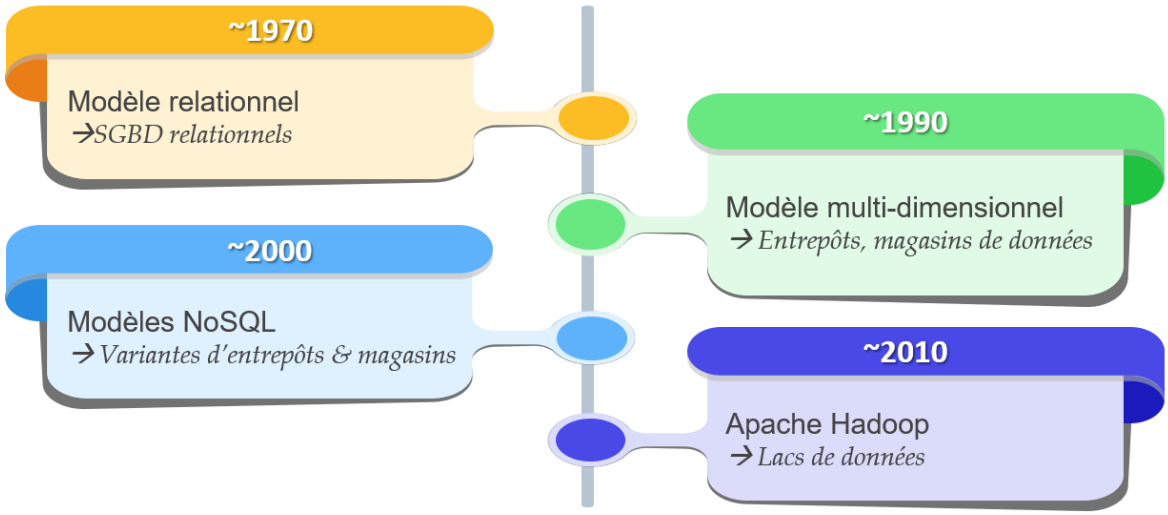


FIGURE 2.11 – Évolution chronologique des systèmes de stockage et décisionnels

confusions demeurent. L'un de ces défis est la conception d'un système de métadonnées efficace pour éviter que le lac ne se transforme en marécage de données. Dans le prochain chapitre, nous analyserons les différentes approches proposées pour y parvenir.

Chapitre 3

Organisation des métadonnées dans les lacs de données

Sommaire

3.1 Introduction

La littérature est quasi-unanime sur le rôle essentiel des métadonnées dans les lacs de données (Diamantini et al., 2018 ; Hai et al., 2016 ; Maccioni & Torlone, 2017). En effet, les lacs de données suivant une approche *schema-on-read*, il n'existe pas de schéma fixe permettant d'interroger les données. L'exploitation des données dépend alors des métadonnées. Au delà de permettre l'interrogation des données et ainsi empêcher le syndrome du marécage de données, les métadonnées forment la base de la gouvernance des données dans les lacs. La gouvernance des données consiste à organiser le cycle de vie des données et à assurer la sécurité ainsi que la qualité des données (Derakhshannia et al., 2020). Pour ce faire, les métadonnées sont organisées dans les lacs de données à travers des systèmes de métadonnées.

La conception d'un lac de données implique alors celle d'un système de métadonnées qui se doit d'être efficace, mais cela n'est pas aisé car il n'existe à ce jour pas de standard d'organisation de ces systèmes de métadonnées, ni même sur les métadonnées à y inclure. Pour donner un aperçu des pratiques actuelles en termes de gestion des métadonnées, nous dressons dans ce chapitre un panorama des tendances en termes de gestion des métadonnées dans les lacs de données.

Plus concrètement, nous proposons dans la première partie de ce chapitre une analyse du concept de métadonnées. Nous y définissons le concept de métadonnées. Nous y présentons également des exemples de métadonnées utilisées et utilisables dans le contexte des lacs de données et, de façon plus générale, dans les systèmes d'information. Dans la deuxième partie de ce chapitre, nous analysons l'organisation des métadonnées dans les lacs de données sous deux aspects distincts. D'une part, nous présentons en détails le fonctionnement de cinq exemples de systèmes de métadonnées de lacs de données proposés dans la littérature. Nous mettons ainsi en lumière des stratégies différentes de gestion des métadonnées en lien avec des cas d'usage précis. Cependant, ces systèmes de métadonnées sont parfois si spécifiques ou peu détaillés qu'ils sont difficilement reproductibles. C'est pourquoi nous proposons d'autre part cinq autres exemples de modèles de métadonnées, qui eux abordent la question de l'organisation des métadonnées dans un cas général. Enfin, nous abordons la problématique de génération des métadonnées dans un quatrième temps.

3.2 Concept de métadonnées

3.2.1 Définition et rôles

Définition

Étymologiquement, le terme « métadonnées » peut être traduit par **« données sur des données »**. Les métadonnées sont donc avant tout des données. Les métadonnées se distinguent toutefois sur deux aspects.

— Les métadonnées viennent en **complément** de données principales. Elles n'existent donc qu'à travers des données qu'elles décrivent.

— Les métadonnées sont **structurées** à un certain point, contrairement aux données qu'elles complètent, qui sont souvent sous une forme brute (Riley, 2017).

Riley (2017) se base notamment sur ces deux caractéristiques pour fournir une définition du concept de métadonnées, que nous adoptons.

Définition 3.2.1. 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.

Il existe une idée reçue qui considère que les métadonnées sont de petite taille, comparées aux données principales. Cela n'est pas vrai, surtout dans le contexte actuel des *big data.* Par exemple, les journaux d'évènements (*logs*) générés par un service peuvent ainsi surpasser les données gérées par le service en termes de volume. De même, la généalogie des données (*lineage*) peut s'accroître de façon importante et ainsi générer d'énormes volumes de métadonnées (Hellerstein et al., 2017).

Métadonnées dans les systèmes d'information

Les métadonnées sont inhérentes à quasiment tous les systèmes d'information que nous utilisons quotidiennement (Riley, 2017). Ainsi, dans les plateformes d'écoute et de visualisation en ligne comme Youtube, Netflix[1] ou Spotify[2] on retrouve des métadonnées comme l'audience (nombre de « vues » sur Youtube), la description, le titre, la date de création, l'auteur, les *likes* et les commentaires associés à un contenu. On y retrouve également des listes de lectures ou albums qui relient les contenus entre eux (vidéos, musiques). On retrouve des métadonnées similaires dans les réseaux sociaux comme Facebook ou Twitter. Les contenus (*posts*) y sont décrits par des horodatages, des *likes* ou des commentaires. Ils sont aussi connectés entre eux par des *hashtags* ou des mentions, qui regroupent des *posts* autour d'un sujet ou d'un utilisateur, respectivement. Dans les moteurs de recherche comme Google[3], des métadonnées sont utilisées pour personnaliser les réponses. Il s'agit entre autres de la localisation de l'utilisateur, de la langue définie sur son navigateur ou encore de l'historique de recherche et de navigation qui accompagne chaque requête.

Métadonnées dans les entrepôts de données

Les métadonnées jouent un rôle important dans les systèmes traditionnels d'entreposage des données (Diamantini et al., 2018). Les métadonnées servent en effet à assurer la traçabilité du processus d'ingestion des données (ETL), d'une part, et du contexte des données elles-mêmes d'autre part (W. H. Inmon, 2005 ; Kimball & Ross, 2013). Les métadonnées associées aux processus ETL permettent d'en surveiller le déroulement à travers des informations comme la date de lancement ou encore l'horodatage des différentes étapes. Ces informations, stockées dans des fichiers *logs*, permettent d'auditer le système pour déceler ou comprendre d'éventuels dysfonctionnements. C'est le même objectif d'audit qui prévaut à l'association de métadonnées aux données d'un entrepôt. Ces

1. https://www.netflix.com/
2. https://www.spotify.com/
3. https://www.google.com/

métadonnées, stockées dans des « dimensions d'audit », permettent en effet de tracer les modifications de valeurs dans les tables de faits et de dimension. Elles peuvent ainsi servir à comprendre des incohérences apparentes dans les données de l'entrepôt.

Métadonnées dans les lacs de données

Dans le contexte des lacs de données, le rôle des métadonnées est encore plus grand. Au delà de la nécessité d'auditabilité, elles répondent à des besoins de reproductibilité des analyses, d'accès aux données, de compréhensibilité des données, d'intégration des données et de limitation des accès (Bhattacherjee & Deshpande, 2018 ; A. Halevy et al., 2016 ; Mehmood et al., 2019 ; Subramaniam et al., 2021).

1. L'**auditabilité** exprime la nécessité de pouvoir détecter et comprendre d'éventuelles incohérences dans les données du lac. Pour y répondre, le système doit tracer l'historique des traitements appliqués aux données.
2. La **reproductibilité des analyses** vient en complément de l'auditabilité. En effet, au delà de rendre le lac de données auditable, la conservation de l'historique des données et traitements garantit le fait que les analyses réalisées dans le lac de données puissent être reproduites à tout moment.
3. L'**accès aux données** représente la nécessité de permettre aux utilisateurs de retrouver facilement et rapidement les données qui les intéressent sur la base d'un ensemble de critères.
4. La **compréhensibilité des données** désigne la nécessite de donner un sens aux données du lac. L'utilisateur doit donc comprendre le contexte associé à ces données (qui, quoi, quand, comment, où, pourquoi, etc.).
5. L'**intégration des données** exprime la nécessité de connecter les données entre elles. Pour ce faire, les données doivent pouvoir être regroupées ou reliées.
6. La **différenciation des accès aux données** désigne la nécessité de contrôler les droits d'accès aux données du lac et d'adapter les formes d'accès à chaque utilisateur ou profil d'utilisateur.

Plusieurs terminologies sont utilisées dans la littérature pour désigner les données de base dans un lac de données. Beheshti et al. (2018) et Eichler et al. (2020) utilisent la notion d'*entité de données* pour représenter de façon générique un document textuel, un document tabulaire, une image, une vidéo ou encore une table relationnelle. Diamantini et al. (2018) et Bagozi et al. (2019) parlent quant-à eux de *sources de données*, tandis que A. Halevy et al. (2016), Maccioni et Torlone (2018) et Ravat et Zhao (2019b) utilisent la notion de *dataset*.

À notre avis, la notion de *dataset* se réfère à un contexte de données structurées, et ne sied donc pas pour représenter la diversité des données potentielles dans un lac de données. L'expression « sources de données » nous paraît tout aussi inadaptée, puisque les sources de données sont par définition extérieures au lac de données. Par conséquent, nous adoptons la notion d'entité de données qui se réfère à un concept classiquement utilisé en modélisation informatique, par exemple avec le modèle Entité-Association (Bruyant & Gateau, 1987). Dans la suite de ce manuscrit, nous utilisons donc les termes « entité de données » pour désigner de façon générique les données de base contenues dans un lac de données.

3.2.2 Typologie fonctionnelle des métadonnées dans les lacs de données

Oram (2015) propose un inventaire des métadonnées utilisables dans un lac de données à travers une typologie, qui est d'ailleurs la plus citée dans la littérature sur les lacs. Cette typologie catégorise les métadonnées du point de vue de leur mode de génération, ainsi que de leur rôle (d'où l'appellation « typologie fonctionnelle »). Elle distingue ainsi trois catégories de métadonnées, à savoir les métadonnées métiers (*business metadata*), les métadonnées opérationnelles (*operational metadata*) et les métadonnées techniques (*technical metadata*).

1. Les **métadonnées métiers** se définissent comme un ensemble de descriptions qui permettent de rendre les données compréhensibles et qui définissent les règles de gestion applicables aux données. Plus concrètement, il s'agit par exemple des noms des attributs, ainsi que des contraintes d'intégrité, qui leur sont applicables. Ces métadonnées sont généralement définies par les utilisateurs métiers lors de l'ingestion des données.
2. Les **métadonnées opérationnelles** sont des informations générées automatiquement lors du traitement des données. Ce sont entre autres des descriptions des données utilisées et produites par des traitements (emplacements, taille des fichiers), ainsi que des informations sur le déroulement des traitements eux-mêmes (nombre d'enregistrements traités, succès ou échec du traitement, etc.).
3. Les **métadonnées techniques** expriment la manière dont les données sont représentées. Cela inclut le format (texte brut, image, document JSON ou XML, etc.) ou encore la structure des données (noms, types, longueurs et contraintes associées aux attributs, etc.). Ces métadonnées sont généralement obtenues à partir des SGBD dans le cas de données structurées, ou encore à l'aide de techniques personnalisées au cours de l'étape de maturation des données.

Cette typologie a été amendée par Diamantini et al. (2018), qui montrent qu'il existe une intersection non nulle entre les trois catégories de métadonnées. En effet, les noms d'attribut peuvent être considérés à la fois comme des métadonnées métiers et techniques. De même, le format des données peut être considéré comme à la fois technique et opérationnel, et ainsi de suite (Figure 3.1).

La relative popularité de la typologie fonctionnelle des métadonnées est sans doute imputable au fait qu'elle s'inspire des catégories de métadonnées des entrepôts de données (Ravat & Zhao, 2019a). Son adoption est donc plus facile et plus naturelle pour les praticiens qui l'utilisent déjà. Elle comporte cependant des faiblesses, à commencer par son imprécision soulignée par Diamantini et al. (2018). De plus, elle occulte des métadonnées importantes comme les connexions entre les données. Une nouvelle typologie de métadonnées plus complète et plus précise nous paraît donc nécessaire pour guider l'identification des métadonnées pertinentes lors de la construction d'un lac de données.

3.3 Implémentations de lacs de données

Dans le but d'illustrer les stratégies possibles d'organisation et d'utilisation des métadonnées dans les lacs de données, nous présentons dans cette partie le fonctionnement de cinq

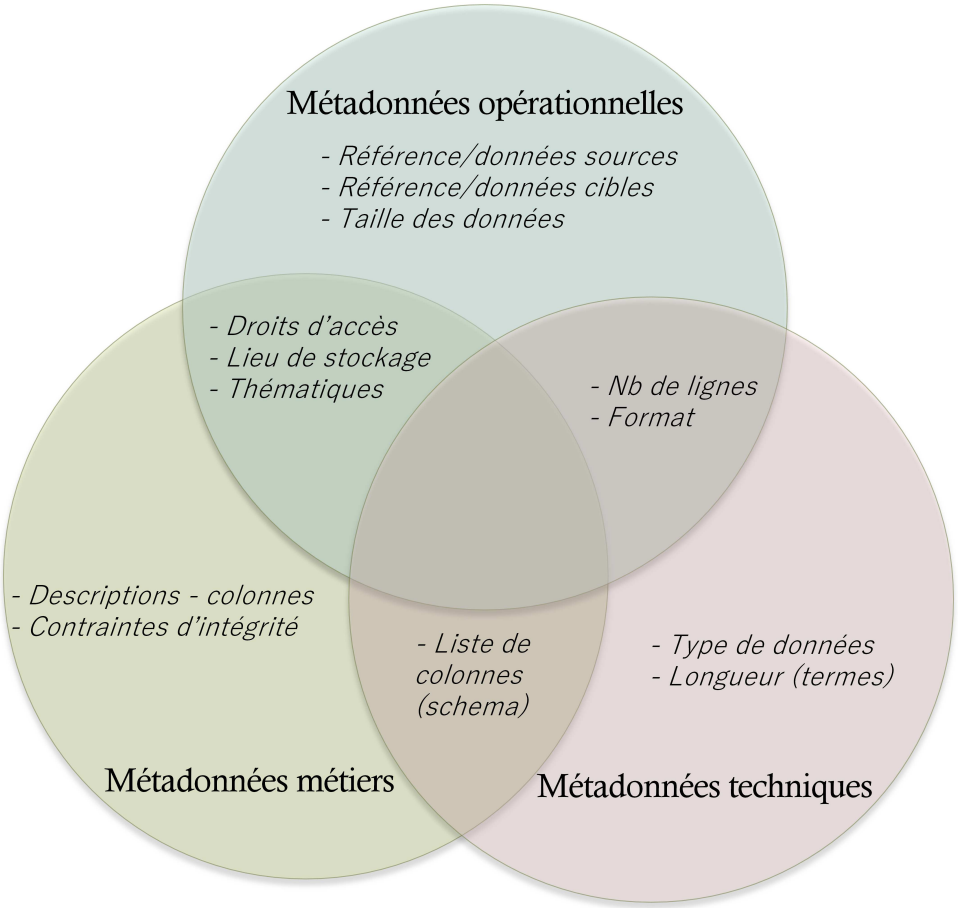


FIGURE 3.1 – Typologie fonctionnelle de métadonnées

implémentations de lacs de données. À travers ces exemples, nous analysons les avantages et inconvénient de différentes approches, à la lumière de cas d'usage concrets.

3.3.1 Lac de données personnelles

Contexte et objectifs

Les interactions entre les services numériques en ligne (réseaux sociaux, boutiques en ligne) et leurs utilisateurs génèrent des données personnelles. La gestion de ces données est généralement laissée aux seuls fournisseurs de services, qui les exploitent à leur guise. C'est dans ce contexte qu'ALREHAMY et WALKER (2015) proposent de redonner aux utilisateurs le contrôle sur leurs données personnelles pour en gérer l'utilisation, la sécurité et la confidentialité. Pour ce faire, les auteurs proposent le concept de lac de données personnelles.

Cela consiste pour un utilisateur à stocker de façon centralisée toute son empreinte digitale (courriels, photos, appels, paiements, etc.). Un tel outil doit ainsi prendre en charge des données structurées, semi-structurées et non-structurées. Au delà de stocker et de contrôler l'accès aux données, le lac de données personnelles doit déduire des connexions sémantiques entre les données, de sorte à permettre des recherches par thématique.

Gestion des métadonnées

Le système de métadonnées proposé est basé sur la théorie des graphes. Chaque entité de données qui arrive dans le lac est automatiquement associée à quatre nœuds stockés dans Neo4J [4].

1. Un **nœud d'identification** permet de regrouper tous les nœuds relatifs à une même entité de données. Il est relié aux autres nœuds (de la même entité de données) par une arête non-orientée.
2. Un **nœud de données brutes** sert à référencer les données brutes (fichiers) associées à l'entité de données stockée.
3. Un **nœud de métadonnées** contient des informations détaillées sur la source des données, le format, les thématiques, etc. Ces informations sont fournies sous la forme d'URI [5] (*Universal Resource Identifier*) reliées à des concepts ontologiques.
4. Un **nœud de données sémantiques** référence les ontologies dont les concepts sont référencés dans le nœud de métadonnées.

Comme l'illustre la Figure 3.2, des arêtes contre-intuitivement appelées « connexions intra » relient les nœuds de métadonnées sémantiquement proches. On peut ainsi connecter des entités de données différentes dans le lac de données personnelles.

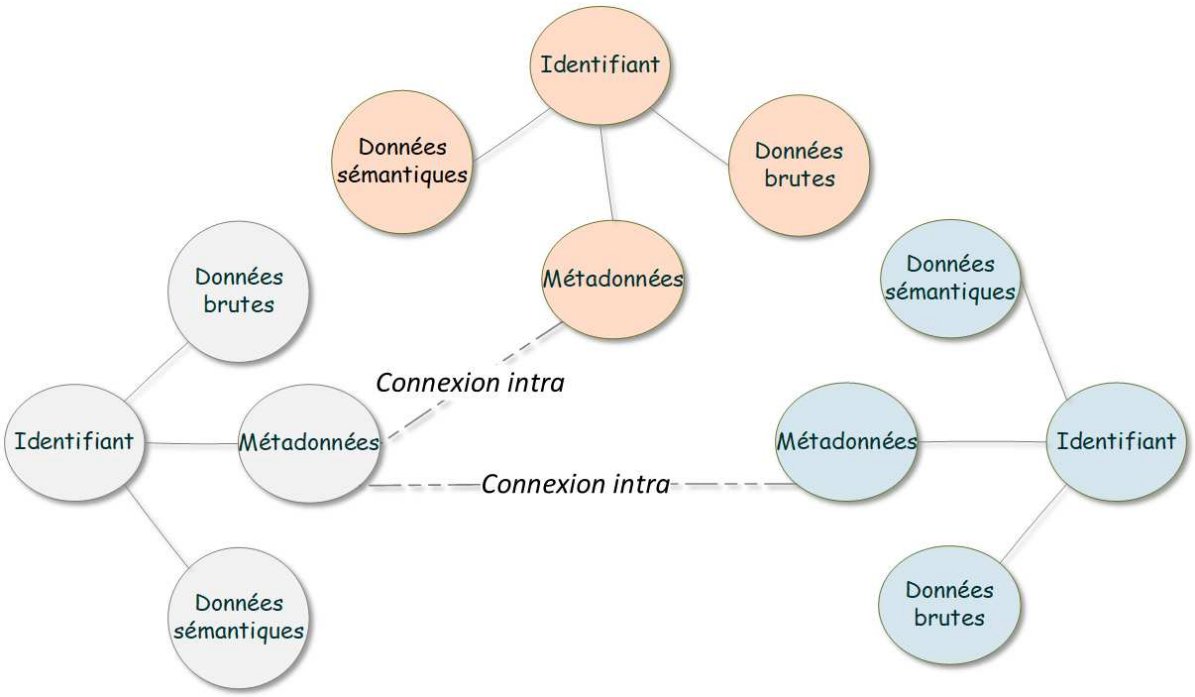


FIGURE 3.2 – Système de métadonnées d'ALREHAMY et WALKER (2015)

4. https://neo4j.com/
5. https://www.w3.org/Addressing/URL/uri-spec.html

Analyses et limitations

Le lac de données personnelles propose une interface d'analyse définie à travers une API REST[6]. Cette interface permet de rechercher des données contenues dans le lac sur la base de mots clés (plus précisément en explorant des catégories définies par des concepts ontologiques). Les expériences des auteurs montrent que le système est efficace et efficient pour des données de taille limitée (jusqu'à 75 Ko). La capacité de passage à l'échelle du système reste donc à démontrer. Une autre limitation du système est qu'il ne supporte pas de données non structurées dans sa version actuelle.

3.3.2 GOODS, le lac de données de Google

Contexte et objectifs

La plupart des grandes entreprises connaissent aujourd'hui une explosion de la quantité de données qu'elles génèrent en interne. Contrairement aux logiciels et scripts dont la gestion est régie par des méthodes standards (versionnement du code, indexation, révisions, tests, etc.), il n'existe pas de techniques largement acceptées pour la gestion des données. Pourtant, de telles techniques sont nécessaires pour éviter le syndrome de silotage des données qui induit des pertes significatives de productivité et souvent une duplication des traitements appliqués sur les données.

Pour répondre à ce besoin, A. Halevy et al. (2016) proposent le lac de données *Google Dataset Search* (GOODS). Il permet de collecter et d'intégrer les données produites en interne par les équipes de Google, tout en travaillant en arrière-plan. Il vise ainsi à rendre les données stockées plus accessibles et plus compréhensibles aux utilisateurs.

Gestion des métadonnées

Pour intégrer les données stockées, GOODS infère automatiquement et continuellement les métadonnées dans un catalogue qui est organisé sous la forme d'un graphe de connaissances, c'est-à-dire un graphe décrivant les relations entre plusieurs concepts. Dans ce cas précis, plusieurs types de nœuds sont utilisés pour représenter les utilisateurs, les équipes de projet, les données, les traitements, etc. Différents types d'arêtes représentent les relations entre ces éléments.

Le catalogue de métadonnées permet ainsi de traduire quatre types d'informations en un graphe de connaissances.

1. Des informations de type **inclusion** permettent de noter le cas d'entités de données incluses dans d'autres.
2. Des informations de **provenance** servent à tracer la généalogie des données (propriétaires, utilisateurs, utilisation dans des jointures, etc.).
3. Des **groupes logiques** permettent de rassembler différentes versions de la même entité de données.
4. Des informations sur la **similarité des contenus** permettent de lier des entités de données dont les valeurs sont proches.

6. Une API REST est une interface de programmation d'application qui permet d'interagir avec des services web.

La Figure 3.3 illustre l'organisation des métadonnées dans le graphe de connaissance de GOODS.

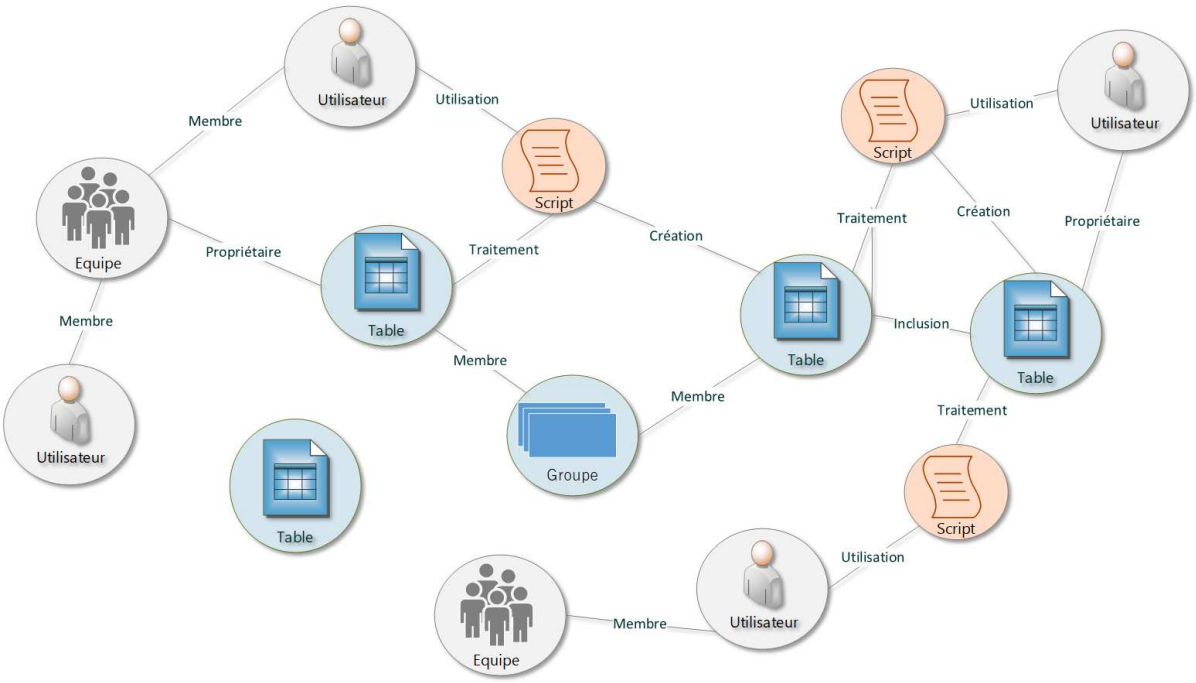


FIGURE 3.3 – Catalogue de métadonnées de GOODS

Analyses et limites

Grâce à son graphe de connaissances et à un système d'indexation, GOODS permet aux utilisateurs de retrouver les données qui les intéressent sur la base de mots clés. Il permet également, grâce aux métadonnées de provenance, de reconstruire l'historique d'une entité de données. Toutes ces analyses passent à l'échelle (GOODS contient environ vingt milliards d'entités de données).

Le lac de données GOODS reste tout de même limité sur deux aspects. D'une part, il se limite à la recherche de données et ne permet pas d'interroger le contenu des entités de données (via SQL, par exemple). D'autre part, GOODS supporte uniquement des données structurées et semi-structurées. Le système exclut donc les données non structurées qui pourtant constituent la majorité des *big data*.

3.3.3 Lac de données en *data vault*

Contexte et objectifs

Ce lac de données fait suite aux travaux de PATHIRANA (2015), qui propose un système de stockage et d'analyse de données patrimoniales constituées d'images, de fichiers XML, d'articles de presse ou encore de livres numériques. Dans ce lac de données, les métadonnées sont organisées suivant un modèle analogue au modèle en étoile des entrepôts de données. Mais cette approche supporte difficilement les évolutions de schéma, à l'image des approches traditionnelles d'entreposage des données.

C'est pourquoi Nogueira et al. (2018) proposent à la place l'utilisation du modèle ensembliste *data vault* pour l'organisation des métadonnées. Les auteurs visent ainsi à apporter plus d'évolutivité et de flexibilité dans le stockage des métadonnées, tout en assurant un traitement rapide des requêtes.

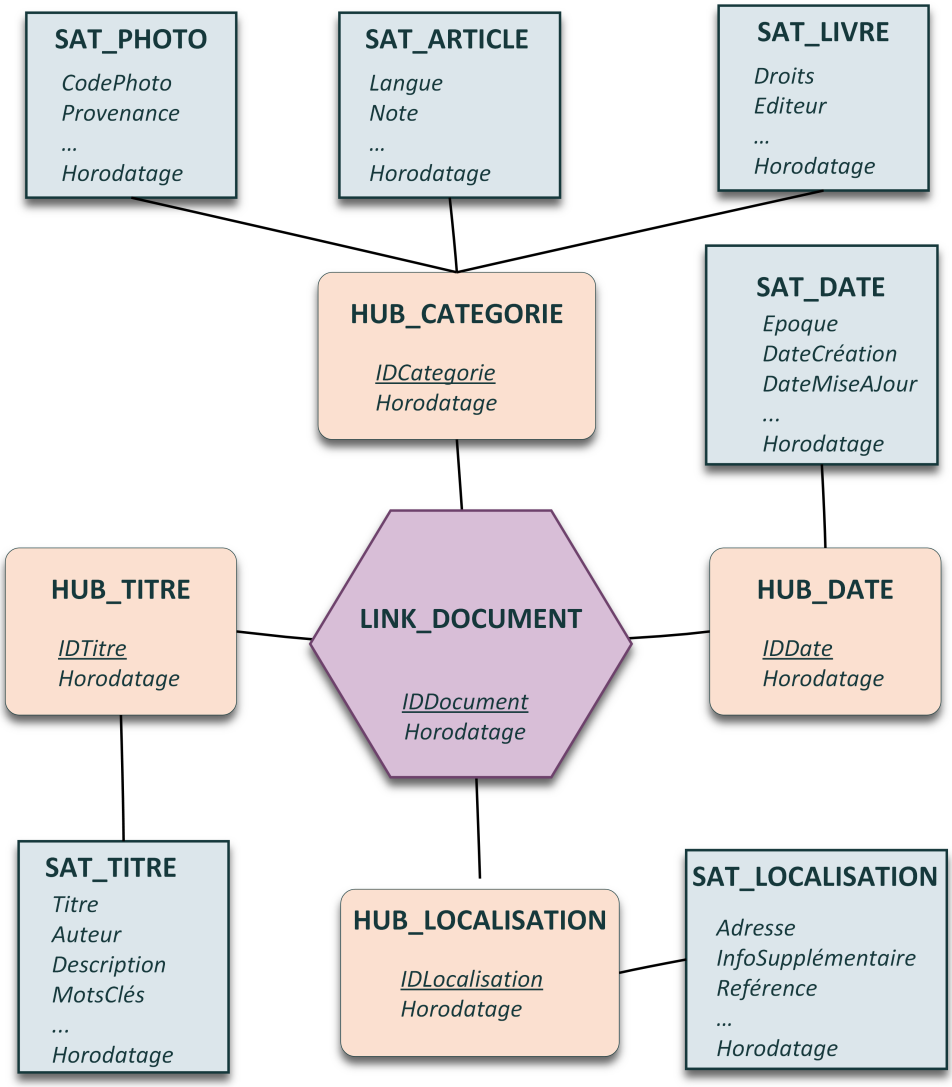


Figure 3.4 – Modélisation des métadonnées en *data vault*

Gestion des métadonnées

Pour rappel, la modélisation en *data vault* implique trois types d'entités (Section 2.3.2).

1. Un ***hub*** représente un concept métier de base, par exemple un client, un produit, une personne, etc.
2. Un ***link*** (lien) traduit une association de deux ou plusieurs *hubs*.
3. Un ***satellite*** contient des informations contextuelles associées à un *hub* ou à un lien. Chaque satellite est attaché à un *hub* ou un lien unique. En revanche, chaque lien ou *hub* est potentiellement associé à plusieurs satellites.

Dans l'implémentation de Nogueira et al. (2018), les métadonnées communes à tous les types de données (titre, catégorie, date de création, emplacement) sont représentées par des *hubs*. Les métadonnées plus spécifiques (langue pour les documents textuels ou éditeur pour les livres) sont quant à elles stockées dans des satellites (Figure 3.4). Dans l'hypothèse ou de nouveaux types d'entités de données arriveraient dans le lac, il est prévu la création de nouveaux satellites qui contiendraient les métadonnées spécifiques à ces entités de données.

Analyses et limites

Ce système de métadonnées permet de retrouver rapidement des entités de données contenues dans le lac à travers un filtrage par catégories (livres, articles de presse, images), par date, ou suivant des mots clés contenus dans le titre. Cela nécessite toutefois des requêtes complexes pour lesquelles les capacités de passage à l'échelle du lac de données restent à démontrer. Une autre limite de cette approche est la non prise en charge d'analyses sur le contenu des entités de données. Il paraît donc nécessaire d'associer des techniques d'analyse d'images et de textes au système proposé pour élargir le spectre des analyses possibles.

3.3.4 CoreKG, un lac de connaissances

Contexte et objectifs

Le système CoreKG (Beheshti et al., 2018) est une extension du lac de données CoreDB précédemment introduit par les mêmes auteurs (Beheshti et al., 2017). CoreDB proposait déjà un système de stockage et d'analyse de données de tous types (structurées, semi-structurées et non structurées) à travers un système d'indexation et d'annotation. CoreDB ne permettait cependant pas d'analyser la provenance des entités de données.

CoreKG lève cette limitation en permettant d'identifier et d'analyser les connexions entre les données du lac. Pour ce faire, il intègre un ensemble de techniques d'enrichissement des données qui permettent d'obtenir non plus seulement un lac de données mais un « lac de connaissances ».

Gestion des métadonnées

Dans CoreKG, le système de métadonnées génère et organise les métadonnées suivant trois techniques complémentaires.

1. Un service d'**indexation** permet de référencer les données brutes dans un index en vue de leur interrogation par mots clés.
2. Un service d'**annotation des données** associe aux données brutes des mots clés ou entités nommées (noms de personnes, localisations, etc.) automatiquement détectés. Il permet également, à l'aide de ressources sémantiques, d'étendre les annotations précédemment obtenues.
3. Un service de **suivi de la provenance** permet quant à lui de tracer les interactions entre les utilisateurs et les données du lac. Il s'appuie sur un graphe de connaissance dont les nœuds représentent des entités (utilisateurs, entités de données), tandis que

les arêtes traduisent des opérations de création, lecture, mise à jour ou suppression. La Figure 3.5 illustre le type de graphe de connaissances utilisé dans CoreKG.

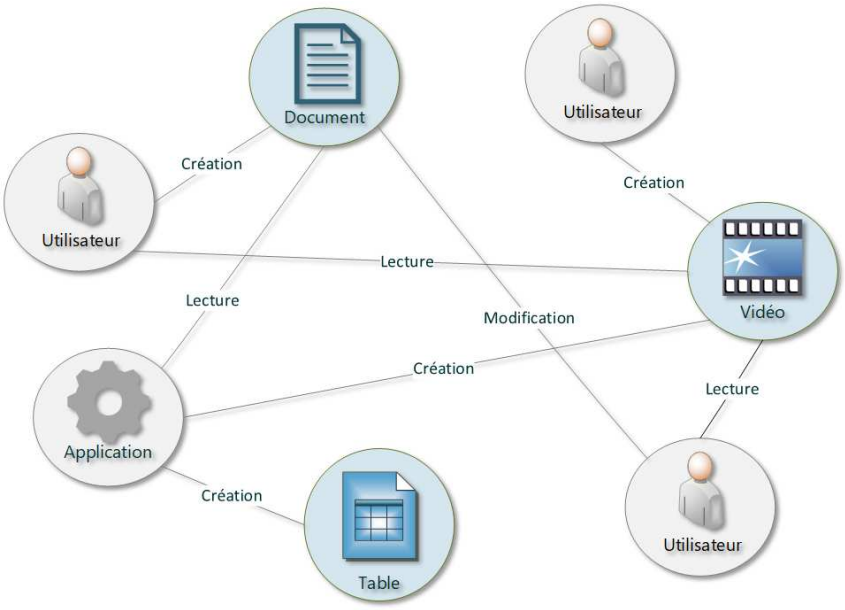


FIGURE 3.5 – Système de suivi de la provenance dans CoreKG

Analyses et limites

À travers son API REST, le système CoreKG supporte une grande panoplie de requêtes, à commencer par l'analyse des connexions entre les entités de données ou encore le suivi de la généalogie des données. Il permet également la recherche par mots clés, aussi bien pour les données structurées que non structurées. Plus spécifiquement pour les données structurées, CoreKG permet d'analyser le contenu des entités de données à travers des requêtes SQL. Toutefois, aucune analyse équivalente n'est possible pour les données non structurées. Une autre limite du systeme CoreKG est l'absence de versionnement des entités de données, qui pourrait occasionner des pertes d'informations lors des opérations de modification.

3.3.5 Lac de données sémantiques

Contexte et objectifs

Dans les villes intelligentes (*smart cities*), de grands volumes de données sont rendues disponibles par les systèmes transactionnels, les objets connectés ou encore les plateformes de données ouvertes. Le concept de lac de données a souvent été proposé comme support d'analyse de ces données. Cependant, la plupart des approches de conception de lacs de données proposent des analyses inadaptées aux utilisateurs métiers. Pour ouvrir l'utilisation du lac à ce type d'utilisateurs, BAGOZI et al. (2019) proposent le concept de lac de données sémantiques. Plus concrètement, les auteurs ont conçu le lac à l'aide de technologies du Web sémantique, de sorte à prendre en charge des opérations de navigation et d'agrégation sur les données du lac.

Gestion des métadonnées

Dans la proposition de Bagozi et al. (2019), le système de métadonnées est généré en deux étapes. Dans un premier temps, des utilisateurs métiers sont chargés d'associer des concepts ontologiques aux données brutes. Il s'agit plus précisément d'annoter chaque attribut d'entité de données avec un concept ontologique. Dans un second temps, des analystes de données créent des indicateurs en liant les concepts ontologiques précédemment définis aux valeurs des données stockées. Les indicateurs sont alors représentés à l'aide d'une ontologie multidimensionnelle où les concepts ontologiques sont traduits en dimensions. L'ensemble des indicateurs est rassemblé dans un graphe d'exploration.

La Figure 3.6 présente un exemple simplifié d'ontologie multidimensionnelle. Elle représente un indicateur global « EnergieBatiment » qui mesure l'energie totale consommée d'un bâtiment. Cet indicateur est exprimé comme la somme de trois sous-indicateurs : « EnergieAscenseurs », « EclairageJardins » et « EclairageEscaliers », qui sont eux-mêmes associés à des dimensions spatiales.

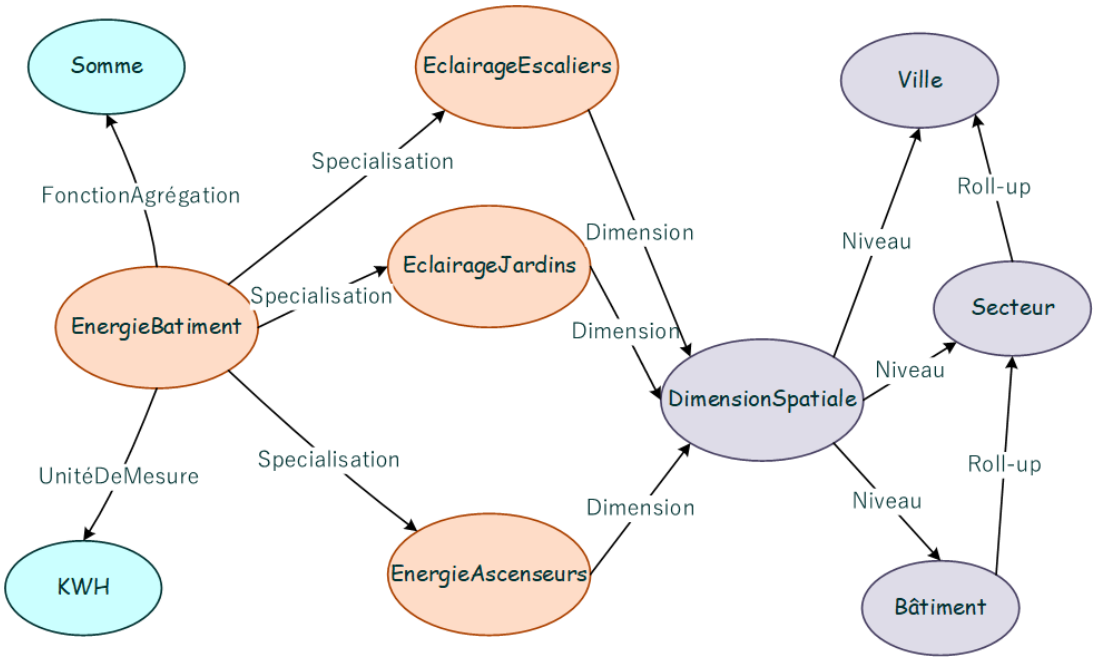


Figure 3.6 – Ontologie multidimensionnelle.

Analyses et limites

Le lac de données sémantiques ainsi obtenu permet aux utilisateurs métiers de naviguer à travers les données stockées et d'agréger les indicateurs définis en fonction de plusieurs dimensions. Les données visibles et interrogeables sont personnalisées par rapport au profil de chaque utilisateur. Ce système est tout de même limité, en dépit de ses avantages d'intuitivité et de personnalisation des analyses. Les accès basés sur les technologies du Web sémantique induisent en effet des problèmes de passage à l'échelle et cette approche ne fait pas exception, comme le reconnaissent les auteurs eux-mêmes. De plus, le système se limite aux données structurées et semi-structurées et nécessite donc d'être réadapté dans un contexte de données non structurées.

3.4 Modèles de métadonnées

Les implémentations de lacs de données présentées précédemment nous ont permis d'illustrer plusieurs cas d'usage et plusieurs stratégies d'organisation des métadonnées. Cependant, les systèmes de métadonnées associés à ces implémentations sont pour la plupart spécifiques et par conséquent difficilement reproductibles en l'état.

C'est pourquoi nous présentons dans cette section cinq exemples de modèles de métadonnées qui, eux, abordent l'organisation conceptuelle des métadonnées tout en étant génériques. La notion de généricité désigne le fait que ces modèles de métadonnées sont suffisamment complets ou flexibles pour s'adapter à plusieurs, si ce n'est à tous les cas d'usage possibles.

3.4.1 GEMMS

Présentation

L'approche de modélisation GEMMS (*Generic and Extensible Metadata Management System*) a été introduite par Quix et al. (2016) pour l'organisation des métadonnées dans les lacs de données. Il vise à décrire les entités de données stockées dans le lac et prend en compte trois catégories de métadonnées.

1. Les **métadonnées propriétés** sont des descriptions atomiques associées aux données sous la formes de couples clés-valeurs. Il s'agit par exemple de la taille du fichier, de son emplacement, de sa date de création, etc.
2. Les **métadonnées structurelles** décrivent la façon dont les données sont organisées dans chaque entité de données et servent principalement à la construction de requêtes. GEMMS modélise ces métadonnées sous forme d'arbres pour les données semi-structurées (documents XML ou JSON) et par une représentation matricielle pour les données structurées.
3. Les **métadonnées sémantiques** sont des annotations ajoutées aux données afin de mieux expliciter le sens de leur contenu. Ces métadonnées viennent en complément des métadonnées propriétés. Elles sont représentées à l'aide d'ontologies.

Dans le modèle conceptuel de GEMMS, les données sont représentées par les concepts de « fichier de données » et « d'unité de données ». Le fichier de données représente un ensemble cohérent de données, par exemple un fichier tabulaire, une base de données, un fichiers XML, etc. (ce que nous désignons de façon générique par la notion d'entité de données). Chaque fichier de données est associé à des métadonnées propriétés ainsi qu'à des métadonnées sémantiques. Les unités de données représentent quant à elles des sous-parties de fichiers de données. Il s'agit par exemple de feuilles de calcul dans le cas de fichiers de tableur. Elles sont également associées à des métadonnées propriétés et sémantiques et, de façon plus spécifique, à des métadonnées structurelles.

La Figure 3.7 illustre l'organisation conceptuelle des données suivant le modèle GEMMS.

Discussion

GEMMS est l'approche pionnière au titre de la modélisation générique des métadonnées dans les lacs de données. À travers les concepts de fichier et d'entité de données, le

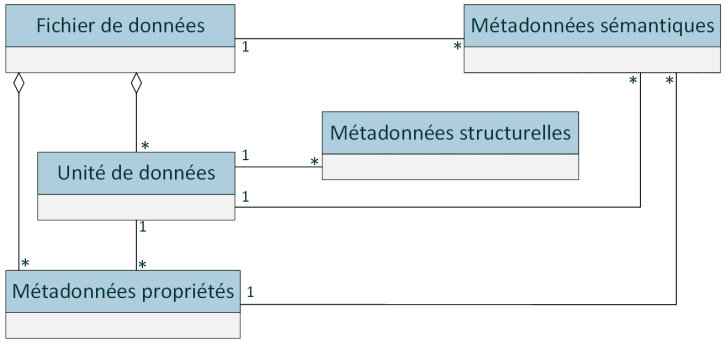


FIGURE 3.7 – Modèle conceptuel de GEMMS

modèle permet de percevoir simultanément deux niveaux de granularité des données. Plus concrètement, cela permet par exemple de gérer à la fois les métadonnées d'une base de données et celles des tables qu'elle contient.

Le modèle GEMMS reste toutefois incomplet sur plusieurs aspects. Il omet en effet les métadonnées définissant les relations entre les données. Or, l'un des principaux usages des lacs de données consiste à détecter ou générer des connexions entre les données. De plus, en exigeant que les fichiers de données soient décrits par un schéma, GEMMS exclut de fait les données non structurées (documents textuels, images, vidéos) qui par définition n'ont pas de schéma. Ce type de données constitue pourtant une partie non négligeable des données exploitables dans les lacs de données.

3.4.2 Ground

Présentation

Le modèle Ground a été introduit par HELLERSTEIN et al. (2017). Il a au départ été proposé indépendamment du concept de lac de données, mais peut tout de même être assimilé à un modèle de métadonnées de lac de données du fait de ses fonctionnalités. Ground vise plus particulièrement à faciliter la recherche de données, d'une part, et à contrôler l'accès aux données, d'autre part. Pour ce faire, le modèle propose une organisation des métadonnées en trois niveaux, basée sur la théorie des graphes.

1. Les **métadonnées applicatives** sont des descriptions qui définissent le sens des données et indiquent comment elles peuvent être utilisées. Il s'agit par exemple de schémas, *tags* ou encodages associés aux données. Elles sont organisées pour chaque *item* (entité de données) à travers un graphe dont un nœud principal représente un *item* et des nœuds secondaires traduisent les métadonnées associées.
2. Les **métadonnées d'utilisation** décrivent comment les données du lac sont créées et utilisées. En d'autres termes, ce sont des informations sur le pédigrée des données. Ces métadonnées sont également organisées via un ensemble de graphes où les nœuds représentent des entités (utilisateurs, groupes, applications) ou des données, tandis que les arêtes traduisent les interactions entre ces entités et les données.
3. Les **métadonnées d'évolution** visent à tracer les changements de schéma ou de contenu des données. Pour ce faire, elles organisent les données à travers un système

de versionnement conçu au niveau logique sous la forme d'un arbre (graphe acyclique et connexe) qui regroupe l'ensemble des versions associées à chaque *item*.

La Figure 3.8 présente le modèle conceptuel de Ground. Par souci d'uniformisation avec les autres modèles présentés, nous le présentons sous la forme d'un diagramme UML, bien qu'il ait été défini et présenté par ses auteurs à travers la théorie des graphes.

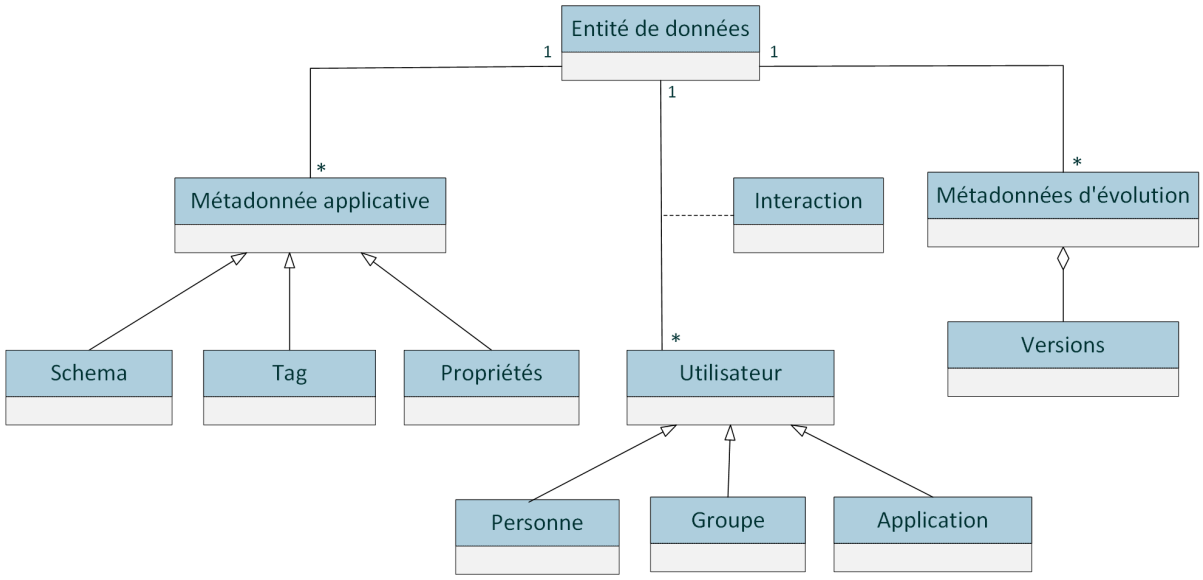


FIGURE 3.8 – Organisation conceptuelle de Ground

Discussion

Le modèle Ground propose une séparation claire du rôle des différentes métadonnées. Il simplifie ainsi la conception du système de métadonnées à travers une organisation en modules quasi-autonomes (qui correspondent aux trois niveaux). L'approche Ground permet également, à la différence du modèle GEMMS, de tracer l'utilisation et le versionnement des données, ce qui contribue à une meilleure gouvernance des données.

Parmi les insuffisances de Ground, on peut noter la non prise en compte de connexions de similarité entre les données. Le modèle Ground peut aussi paraître flou dans la mesure où la séparation entre les métadonnées d'évolution et les métadonnées d'utilisation n'est pas évidente. En effet, ces deux catégories de métadonnées semblent avoir le même rôle, à savoir décrire la généalogie des données. Par conséquent, leur séparation peut induire des confusions.

3.4.3 Modèle de DIAMANTINI et al. (2018)

Présentation

Ce modèle basé sur la typologie fonctionnelle des métadonnées (Section 3.2.2) se focalise sur une organisation des relations entre les données du lac. Pour ce faire, DIAMANTINI et al. (2018) proposent une modélisation sous forme de graphes où les nœuds représentent des « objets », tandis que les arêtes représentent les relations entre eux. La notion d'objet renvoie à la fois à des ensembles cohérents de données, d'une part (fichiers tabulaires,

documents JSON, tables relationnelles), et des descriptions associées à ces ensembles de données, d'autre part (*tags*, noms d'attributs, etc.). Nous les qualifions d'objets complexes et d'objets simples, respectivement.

Deux catégories de relations sont traduites par les arêtes du modèle de Diamantini et al. (2018).

1. Les **relations structurelles** représentent un rapport d'inclusion entre des objets complexes et des objets plus atomiques. Elles sont représentées par le label *contains*. Une telle relation peut par exemple servir à associer des *tags* (objets atomiques) à un fichier vidéo (objet complexe).
2. Les **relations de similarité** permettent de connecter deux objets atomiques. Elles représentent une similarité sémantique obtenue à l'aide d'une ontologie ou une similarité syntaxique.

La Figure 3.9 illustre le modèle conceptuel proposé par Diamantini et al. (2018).

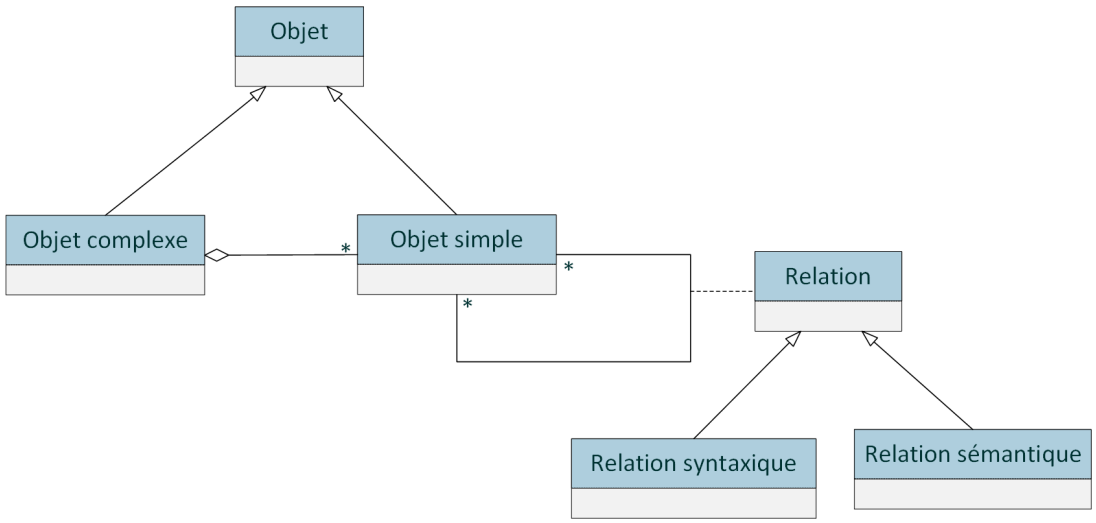


Figure 3.9 – Organisation conceptuelle du modèle de Diamantini et al. (2018)

Discussion

Le modèle de Diamantini et al. (2018) dépasse les limites des modèles GEMMS et Ground en prenant en compte les relations entre les données. Il permet ainsi de détecter des connexions entre des données de tous types (structurées, semi-structurées, non structurées). Il permet aussi de traiter simultanément plusieurs niveaux de granularité des données à travers la dualité objets complexes/objets atomiques.

Comme les modèles précédents, celui de Diamantini et al. (2018) possède des limites, à commencer par la non prise en compte des métadonnées opérationnelles (*cf.* typologie fonctionnelle, Section 3.2.2). De ce fait, le modèle ne permet pas de gérer la généalogie des données. De façon plus générale, ce modèle est tellement centré sur la modélisation des relations entre les données, qu'il semble ne pas prendre en compte les métadonnées propriétés basiques comme le titre, les horodatages de création ou de modification, etc. (Eichler et al., 2020).

3.4.4 Modèle de Ravat et Zhao (2019b)

Présentation

À travers ce modèle, Ravat et Zhao (2019b) visent à conformer la gestion des métadonnées dans les lacs de données avec les architectures en zones (Section 4.2.1). Le modèle proposé se focalise ainsi sur le suivi du cycle de vie des données. Pour construire leur modèle, Ravat et Zhao (2019b) distinguent deux catégories de métadonnées, à savoir les métadonnées intra et inter.

1. Les **métadonnées intra** sont directement associées à une unique entité de données dans le lac. Ce sont entre autres des informations sur la généalogie des données, le schéma, les droits d'accès aux données, etc.
2. Les **métadonnées inter** décrivent quant à elles des relations entre plusieurs entités de données. Il s'agit plus concrètement de relations de similarité, d'inclusion ou de filiation entre deux entités de données, ou encore l'appartenance à un même groupe.

Dans son organisation conceptuelle, le modèle représente chaque entité de données par le concept de *dataset* et des spécialisations de ce concept (*dataset* externe pour des données extérieures au lac, *dataset* interne pour des données à l'intérieur du lac, *dataset* semi-structuré ou non structuré, etc.).

À ces concepts de base s'ajoutent les « relations » qui définissent des connexions entre les données, et les *tags* qui permettent de regrouper les données. Enfin, les interactions entre les entités (utilisateurs et applications) et les données du lac sont tracées par les concepts « accès », « ingestion » et « traitement », qui correspondent à des opérations de lecture, d'ingestion et de transformation de données, respectivement.

La Figure 3.10 résume le modèle conceptuel proposé par Ravat et Zhao (2019b).

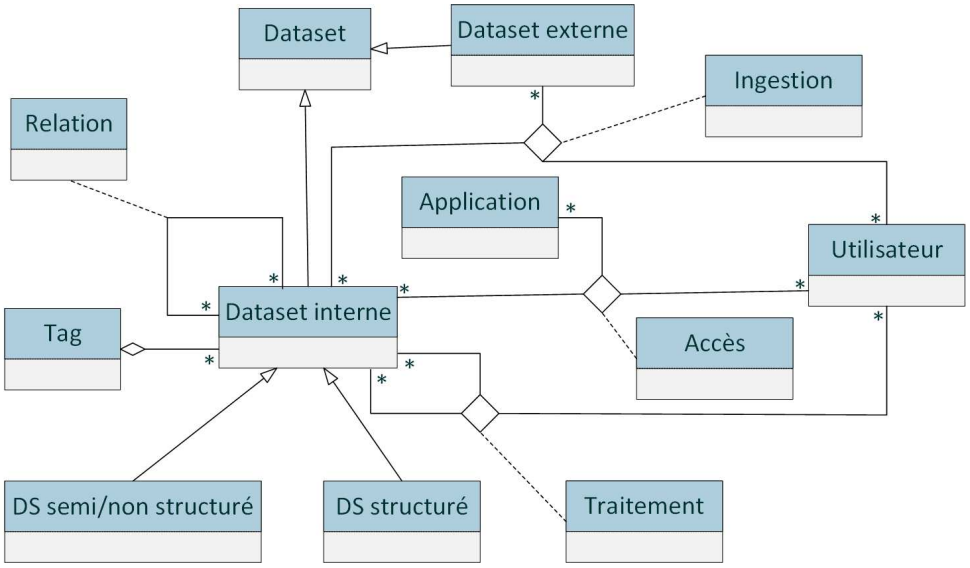


Figure 3.10 – Organisation conceptuelle du modèle de Ravat et Zhao (2019b)

Discussion

Le modèle de Ravat et Zhao (2019b) est l'un des premiers, après celui de Diamantini et al. (2018), à supporter tous types de données (structurées, semi-structurées et non structurées). De plus, il propose comme Ground une différenciation du rôle des différentes catégories de métadonnées et prend en compte les connexions entre les données.

Cependant, de nouveau comme Ground, l'approche de Ravat et Zhao (2019b) sépare les métadonnées liées à la filiation des données de celles relatives à son historique, alors que ces informations sont intrinsèquement liées, ce qui peut engendrer des confusions. Une autre limite notable de ce modèle réside en l'utilisation d'un seul et même concept pour représenter des données non structurées et semi-structurées, qui nous semblent partager en effet plus de points communs avec les données structurées (attributs, schéma). Par conséquent, à défaut d'être représentées à part, elles devraient être assimilées aux données structurées. Enfin, ce modèle se prête mal à l'analyse automatisée du contenu des données (via SQL, par exemple), ce qui le rend peu compatible à une utilisation par des utilisateurs métiers.

3.4.5 HANDLE

Présentation

Ce modèle proposé par Eichler et al. (2020) a pour but de remédier au manque de flexibilité et à la relative complexité des modèles préexistants. HANDLE (*Handling metAdata maNagement in Data LakEs*) vise ainsi à prendre en charge un maximum de cas d'usage de modélisation des métadonnées dans les lacs de données à travers un nombre minimal de concepts. Ces concepts sont séparés en deux catégories dans le modèle conceptuel de HANDLE.

Les concepts du ***core model*** représentent les entités de données (données brutes) et les métadonnées de base (Figure 3.11). Dans HANDLE, les entités de données peuvent être de niveaux de granularité hétérogènes. Par exemple, une entité de données peut représenter une table relationnelle, alors qu'une autre représente un n-uplet. Pour exprimer cette hétérogénéité de niveaux de granularité, HANDLE prévoit des relations de type « inclusion » entre les entités de données, en plus des liens de type « similarité ».

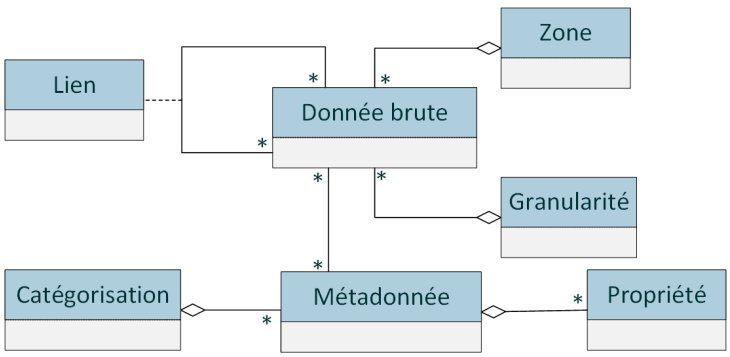


Figure 3.11 – Modèle conceptuel (*core model*) de HANDLE

Trois **extensions** viennent compléter le *core model* à travers des métadonnées parti-

culières. Une première extension intitulée « indicateur de granularité » est associée aux données brutes pour en représenter le niveau de granularité. De même, « l'indicateur de zone » représente le dégré de raffinement des données (données brutes, données raffinées, etc.). Enfin, le concept de « catégorisation » permet de définir plusieurs catégories de métadonnées.

Discussion

Le modèle HANDLE adopte un haut niveau d'abstraction qui lui permet de se conformer à la plupart des cas d'usage définis dans les modèles précédents. Il peut ainsi organiser les données en zones comme le modèle de Ravat et Zhao (2019b) ou gérer simultanément plusieurs niveaux de granularité à l'image du modèle GEMMS.

HANDLE reste toutefois incomplet, dans la mesure où la catégorisation se limite aux métadonnées. Pourtant, divers types de catégorisation sont communément appliquées aux données d'un lac, au delà du niveau de granularité et de la zone (type de données, langue, etc.). De plus, le haut niveau d'abstraction adopté par HANDLE relègue la tâche d'identification des métadonnées pertinentes aux utilisateurs. De ce fait, les utilisateurs doivent recourir à d'autres modèles de métadonnées plus explicites pour les guider dans l'instanciation du modèle de métadonnées.

3.5 Extraction et génération des métadonnées

Une fois un système de métadonnées construit, il doit être alimenté par des instances de métadonnées, provenant des données brutes ou de sources externes. Dans cette section, nous faisons l'inventaire des principales approches utilisées à cet effet.

3.5.1 Technologies et outils pour l'extraction des métadonnées

La plupart des outils qui servent à l'ingestion des données dans un lac sert également à l'extraction de métadonnées (Section 4.3.1). Suriarachchi et Plale (2016) utilisent ainsi Apache Flume [7] pour tracer la provenance des données dans leur lac. De façon analogue, certaines API, en permettant le transfert des données vers le lac, fournissent aussi des métadonnées relatives à ces données. C'est le cas de l'interface d'extraction de données fourni par la plateforme de données ouvertes CKAN [8] (Terrizzano et al., 2015).

D'autres technologies sont plus spécifiques à la problématique d'extraction des métadonnées. Par exemple, Apache Tika [9] permet de détecter automatiquement le type MIME (format), la langue et d'autres propriétés relatives à un document (Quix et al., 2016). D'autres outils comme Open Calais [10] et l'API Alchemy d'IBM [11] permettent également d'enrichir

7. https://flume.apache.org/
8. https://ckan.org/
9. https://tika.apache.org/
10. http://www.opencalais.com
11. https://www.alchemyapi.io/

les lacs de données à travers la détection d'entités nommées ou d'événements (Farid et al., 2016).

3.5.2 Méthodes de génération de métadonnées

La spécificité des différentes implémentations de lacs de données oblige souvent à recourir à des algorithmes *ad hoc* pour la génération de certaines métadonnées. Ainsi, plusieurs approches ont été utilisées dans la littérature pour ajouter des descriptions aux données brutes ou inférer des connexions entre les données.

Association de descriptions

Les métadonnées de type « description » sont communément utilisées pour rendre les données du lac compréhensibles, d'une part, et faciliter leur interrogation, d'autre part. Ainsi, des méthodes automatiques ont été proposées pour déduire le schéma des données brutes. En effet, les données étant intégrées dans le lac sans schéma prédéfini, cette détection *a posteriori* de la structure des données est alors indispensable pour en permettre l'interrogation.

Parmi les techniques proposées à cet effet, on peut citer les approches de Klettke et al. (2017) et de Hamadou et al. (2018). Dans l'approche de Hamadou et al. (2018), les différents chemins d'accès à chaque attribut sont répertoriés dans un dictionnaire, dans le but d'assister les utilisateurs lors de la construction de requêtes. L'approche proposée par Klettke et al. (2017) va plus loin en traçant à travers le temps les évolutions du schéma d'une collection de données semi-structurées.

Pour rendre les données plus compréhensibles, Nargesian et al. (2018a) proposent de transférer des annotations depuis d'autres lacs de données. Plus concrètement, cela consiste en trois étapes. D'abord, des entités de données déjà annotées sont extraites de plateformes de données ouvertes (Open Calais, Wikipedia, etc.) et regroupées selon leurs annotations. Ces entités de données servent ensuite à entraîner des algorithmes de classification qui permettent d'établir une correspondance entre chaque annotation (*tag*) et un groupe de termes ou de valeurs. Enfin, les entités de données à annoter sont tour à tour soumises à chaque classifieur, qui détermine si le contenu des entités de données correspond au *tag* à reconnaître. Si c'est le cas, l'entité de données est annotée avec le *tag* correspondant (Figure 3.12).

Inférence de relations

La détection de relations entre les données est une autre problématique majeure de génération des métadonnées dans les lacs. Dans le cas des données tabulaires, Singh et al. (2016) utilisent un modèle bayesien pour déduire des possibilités de jointures entre des attributs appartenant à des entités de données différentes. Dans le même but de détecter des attributs similaires, Bogatu et al. (2020) et Fernandez et al. (2018) appliquent quant à eux des mesures statistiques entre le contenu des attributs : statistique Kolmogorov-Smirnov (Justel et al., 1997), similarité cosinus (Allan et al., 2000) ou similarité de Jaccard (Ivchenko & Honov, 1998). Ces approches restent cependant limitées face à des données moins structurées et plus hétérogènes.

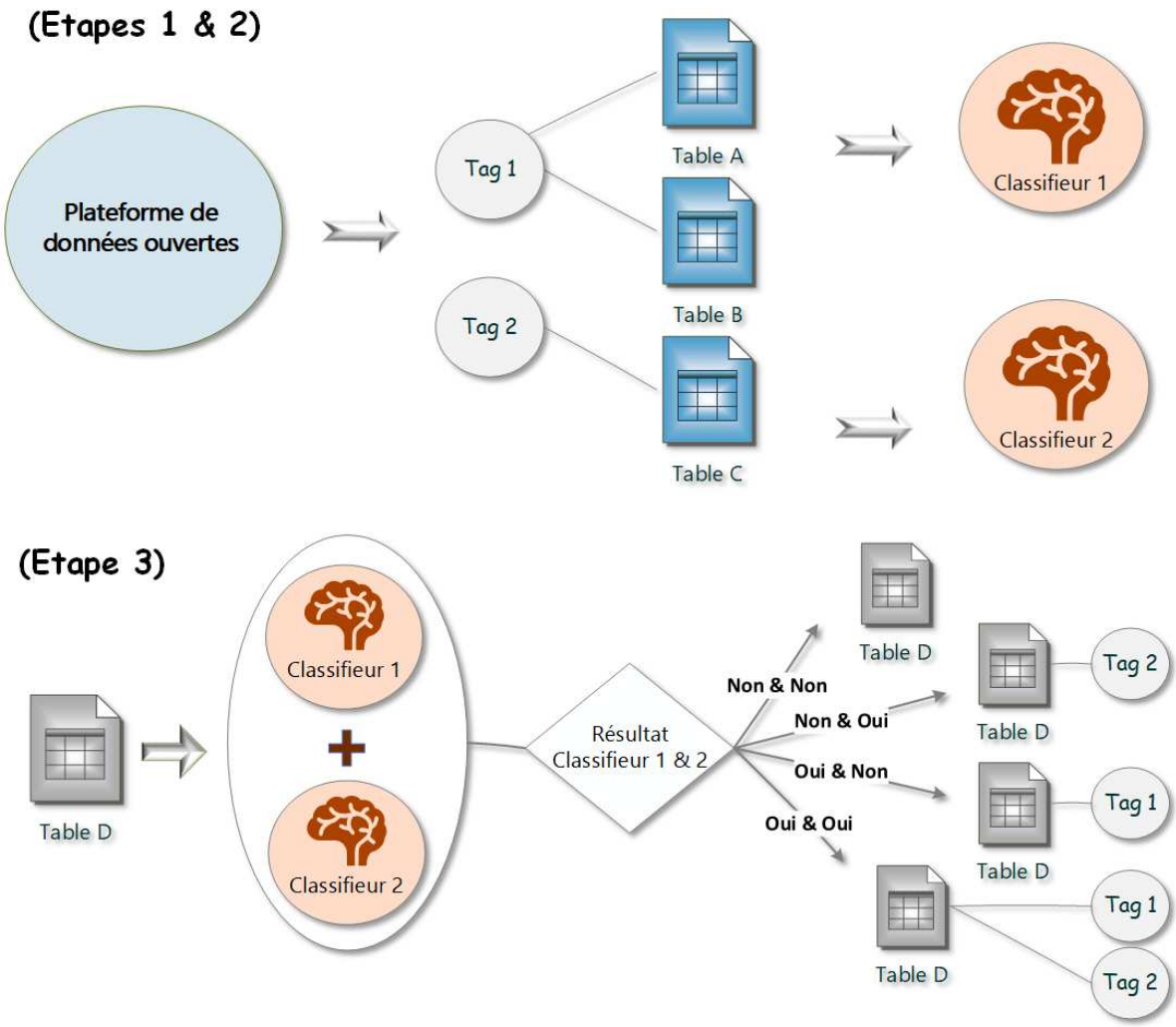


FIGURE 3.12 – Approche de transfert de *tags* de NARGESIAN et al. (2018a)

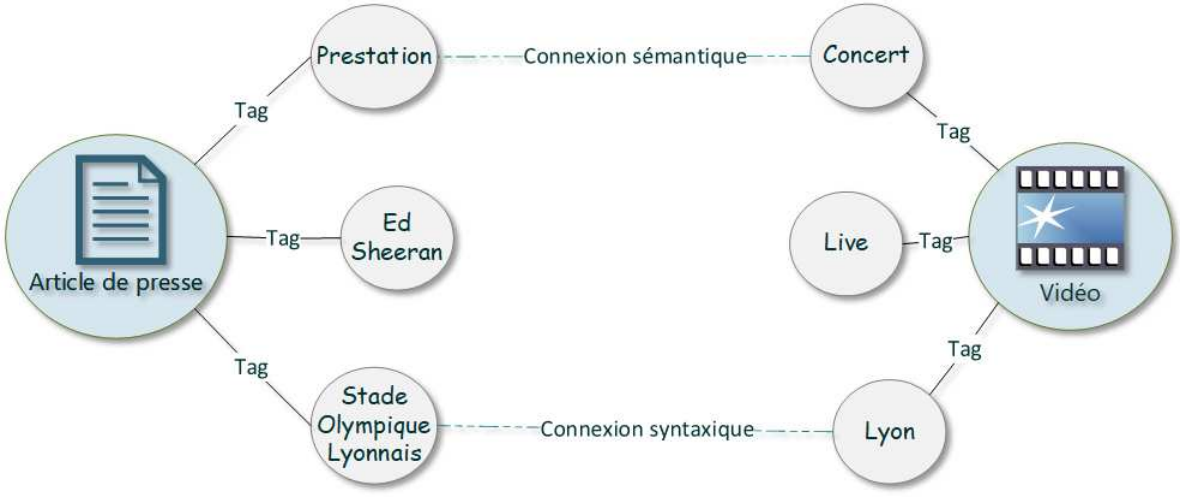


FIGURE 3.13 – Approche de génération de connexions de DIAMANTINI et al. (2018)

Dans de tels cas, l'approche proposée par DIAMANTINI et al. (2018) est plus adaptée. Il s'agit d'un processus en deux étapes, dont la première consiste à extraire des mots clés représentatifs de chaque entité de données dans le lac (*tags* pour des images ou vidéos, noms des colonnes pour des données tabulaires, entités nommées pour des documents textuels, etc.). Dans la deuxième étape, les mots clés précédemment extraits sont comparés deux à deux à l'aide de la distance N-Gram (KONDRAK, 2005) ou d'une ontologie, puis connectés lorsqu'ils sont fortement similaires (syntaxiquement ou sémantiquement). On obtient ainsi des connexions entre des données pourtant de formats divers (Figure 3.13). Une approche similaire est proposée par CHANIAL et al. (2018), dans un contexte différent, mais fortement analogue à celui des lacs de données. Dans leur approche, les mots clés sont « connectés » en utilisant la distance de Jaro (COHEN et al., 2003).

3.6 Conclusion

Comme nous l'avons souligné dans ce chapitre, les métadonnées jouent un rôle clé dans les lacs de données. En plus de rendre les données auditables, elles sont utilisées pour assurer la reproductibilité des analyses, et faciliter l'interrogation des données. Les métadonnées servent aussi à donner du sens aux données brutes, à connecter les données et à contrôler leur utilisation.

Plusieurs approches ont été proposées dans la littérature pour organiser les métadonnées dans les lacs. Nous avons ainsi présenté cinq techniques différentes, orientée chacune vers un cas d'usage précis. Ces implémentations de lacs de données permettent de percevoir concrètement l'utilité et les modes d'utilisation des métadonnées dans les lacs. Les implémentations de lacs de données restent toutefois difficilement reproductibles, car la plupart d'entre elles sont spécifiques au cas d'usage, ou alors insuffisamment détaillées. C'est pourquoi nous avons également présenté cinq modèles de métadonnées plus détaillés et orientés vers l'organisation conceptuelle des métadonnées et des données.

Malgré leur vocation de généricité, aucun des modèles de métadonnées de la littérature ne permet de prendre en charge l'intégralité des cas d'usage possibles. Même HANDLE (EICHLER et al., 2020), qui est le plus complet et le plus récent d'entre eux, demeure incomplet (Section 3.4.5). Des modèles de métadonnées plus génériques et plus complets sont donc encore nécessaires.

Chapitre 4

Architectures, technologies et méthodes applicables aux lacs de données

Sommaire

4.1 Introduction

À l'instar de plusieurs autres systèmes informatiques (et même au delà), la mise en œuvre effective d'un lac de données passe par sa structuration en différents composants de base. Le terme « architecture » est communément utilisé pour représenter l'organisation de ces composants, ainsi que les interactions qu'ils ont entre eux.

Dans l'optique d'étudier les aspects opérationnels d'implémentation des lacs de données, nous proposons dans la première partie de ce chapitre une synthèse des principales approches architecturales associées aux lacs de données. Nous analysons à la fois l'organisation interne des lacs de données, mais aussi leur positionnement dans l'architecture globale du système décisionnel, notamment vis-à-vis des entrepôts de données.

Une fois l'architecture définie, sa mise en œuvre et son fonctionnement effectifs requièrent le concours d'un ensemble de technologies et de méthodes. Ainsi, dans la deuxième partie de ce chapitre, nous faisons l'inventaire des approches techniques et technologiques adaptées aux principales fonctions des lacs de données, à savoir l'ingestion des données, le stockage, l'extraction des métadonnées, l'accès aux données et la surveillance (*monitoring*) du système.

4.2 Architectures associées aux lacs de données

4.2.1 Architectures internes des lacs de données

Dans la littérature, les architectures internes des lacs de données sont communément divisées en deux catégories : l'architecture en zones et l'architecture en bassins de données (*data ponds*). Dans l'architecture en zones, les composants dépendent du niveau de maturité des données, alors qu'ils suivent le format des données (données structurées, données textuelles, données de capteurs...) dans l'architecture en bassins (Giebler et al., 2019 ; Ravat & Zhao, 2019b). Cependant, cette catégorisation n'est pas si nette de notre point de vue. L'architecture en bassins peut en effet être considérée comme une variante d'architecture en zones, car l'emplacement des données y dépend aussi de leur niveau de raffinement. Nous proposons donc dans cette section, une catégorisation alternative à la typologie « bassins - zones ». Nous distinguons ainsi trois principales approches architecturales utilisées dans les lacs de données : les architectures fonctionnelles, les architectures en zones et les architectures hybrides.

Architectures fonctionnelles

Les architectures fonctionnelles définissent les composants du lac de données par rapport à un ensemble de fonctions de base. Les fonctions de base typiquement incluses dans un lac de données sont les suivantes (LaPlante & Sharma, 2016).

1. L'**ingestion des données** consiste à transférer les données dans le lac de données à partir de sources diverses.
2. Le **stockage des données** sert à faire persister les données brutes et raffinées, ainsi que les métadonnées du lac.

3. Le **traitement des données** est un ensemble de méthodes et moyens permettant de transformer ou d'affiner les données du lac.
4. L'**accès aux données** permet d'interroger le lac de données dans le but d'y trouver ou d'en extraire des données brutes ou raffinées.

Les architectures de Quix et Hai (2018) et de Mehmood et al. (2019) peuvent être considérées comme fonctionnelles. Leurs composants correspondent en effet aux fonctions de base d'un lac de données, à savoir l'ingestion des données, le stockage, le traitement et l'accès aux données. Nous notons cependant une variation de dénomination des composants. Mehmood et al. (2019) évoquent ainsi une couche d'exploration et d'analyse et une couche de visualisation qui correspondent aux fonctions de traitement et d'accès aux données, respectivement. Quix et Hai (2018) utilisent plutôt des couches de transformation des données et d'interactions.

La Figure 4.1 présente un exemple d'architecture fonctionnelle, inspirée de celle de Quix et Hai (2018).

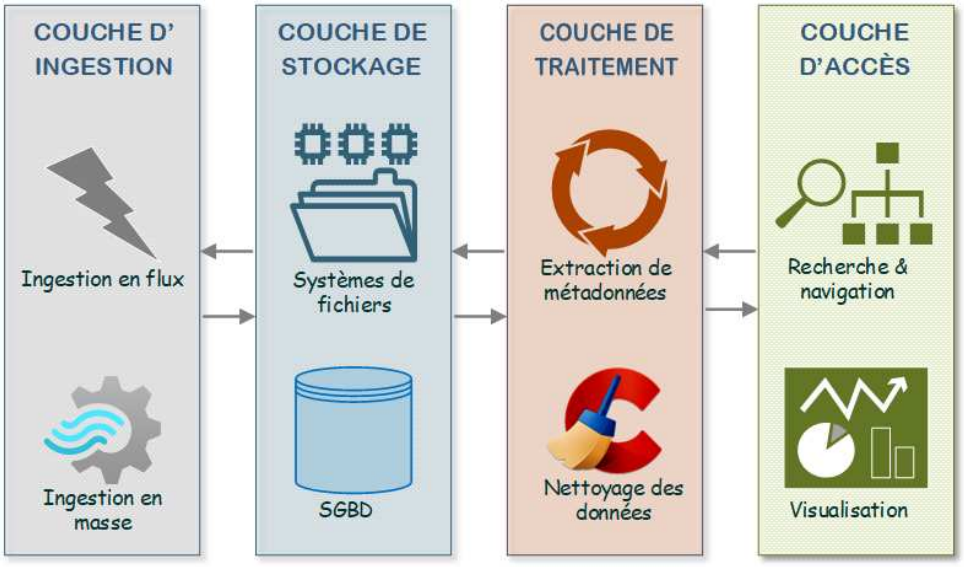


Figure 4.1 – Exemple d'architecture fonctionnelle

Au delà du nommage des composants, certaines architectures fonctionnelles se différencient par le nombre et la spécificité des fonctions implémentées. C'est typiquement le cas de l'architecture Lambda appliquée aux lacs de données (John & Misra, 2017). L'architecture Lambda est à l'origine une architecture générique de gestion des *big data*, indépendante du concept de lac de données. Sa principale caractéristique est de proposer des couches de traitements spécifiques aux données en flux, d'une part, et aux données en lots, d'autre part. John et Misra (2017) proposent une implémentation de l'architecture Lambda en sept couches dans le contexte des lacs de données (Figure 4.2).

1. La **couche d'acquisition des données** permet d'extraire les données de leurs différentes sources.
2. La **couche de messagerie** assure la traçabilité des données lors de leur ingestion dans le lac.

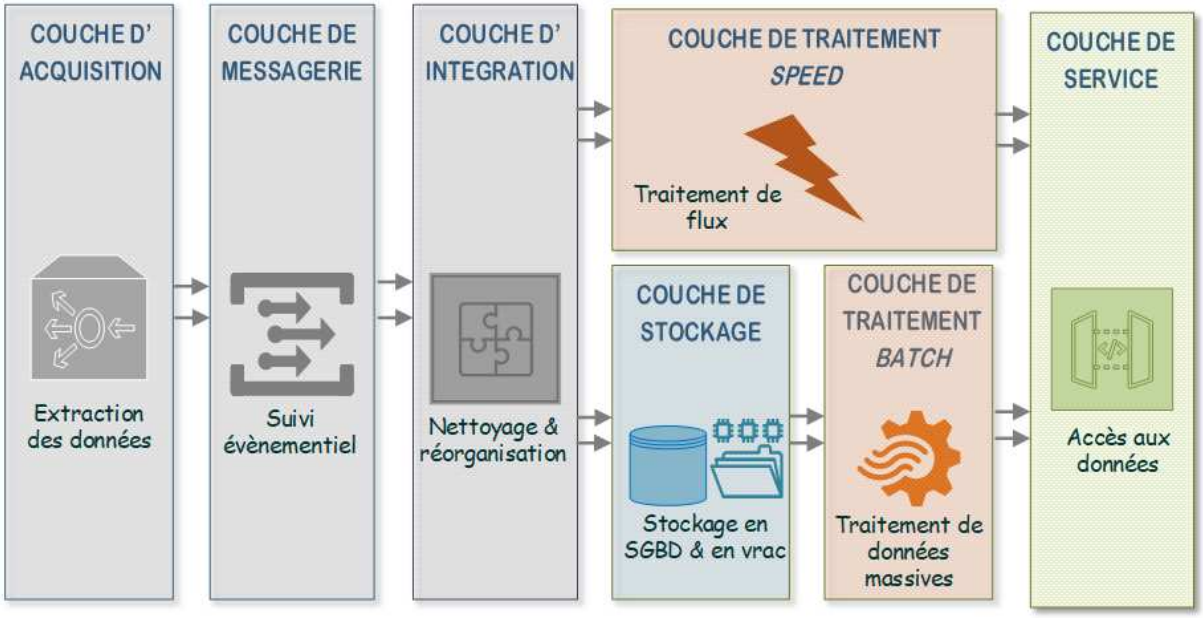


FIGURE 4.2 – Architecture Lambda de JOHN et MISRA (2017)

3. La **couche d'intégration des données** sert à prétraiter les données extraites en vue de leur stockage.
4. La **couche de stockage** permet de faire persister les données à travers des systèmes de fichiers ou des SGBD.
5. La **couche de traitement *batch*** permet de traiter les données en lots (*bulk data*) une fois qu'elles sont stockées dans le lac.
6. La **couche de traitement rapide** est dédiée au traitement en temps réel des données en flux.
7. La **couche de service** fournit un accès aux données du lac à travers un ensemble de protocoles et d'API.

Architectures en zones

Ce sont des architectures de lacs de données où les composants sont définis en fonction du niveau de raffinement des données. Ces composants sont communément appelés « zones ». Par exemple, le lac de données de Zaloni (LAPLANTE & SHARMA, 2016) adopte une architecture en sept zones (Figure 4.3).

1. La **zone de chargement et de transit des données** prend en charge les données en cours d'ingestion.
2. La **zone de données brutes** contient les données brutes nouvellement intégrées dans le lac de données.
3. La **zone des raffinement** sert à nettoyer, agréger et normaliser les données.
4. La **zone de données fiables** est celle où les données sont transférées une fois raffinées. C'est dans cette zone que la fiabilité des données est vérifiée.
5. Le **bac à sable** est dédié à des analyses exploratoires des données issues de la zone de données fiables, effectuées par des *data scientists*.

6. La **zone de consommation** sert à des analyses plus systématiques, notamment effectuées par des utilisateurs métiers.
7. La **zone de gouvernance** permet enfin de gérer, contrôler et gouverner le cycle de vie, la qualité et la sécurité des données.

Il ne s'agit là que d'une approche architecturale en zones parmi plusieurs, mais ces architectures diffèrent en effet essentiellement par le nombre et par les caractéristiques des zones (Giebler et al., 2019). Ainsi, certaines d'entre elles proposent une zone de transit (LaPlante & Sharma, 2016 ; Tharrington, 2017 ; Zikopoulos et al., 2015) contrairement à d'autres (Hai et al., 2016 ; Ravat & Zhao, 2019b).

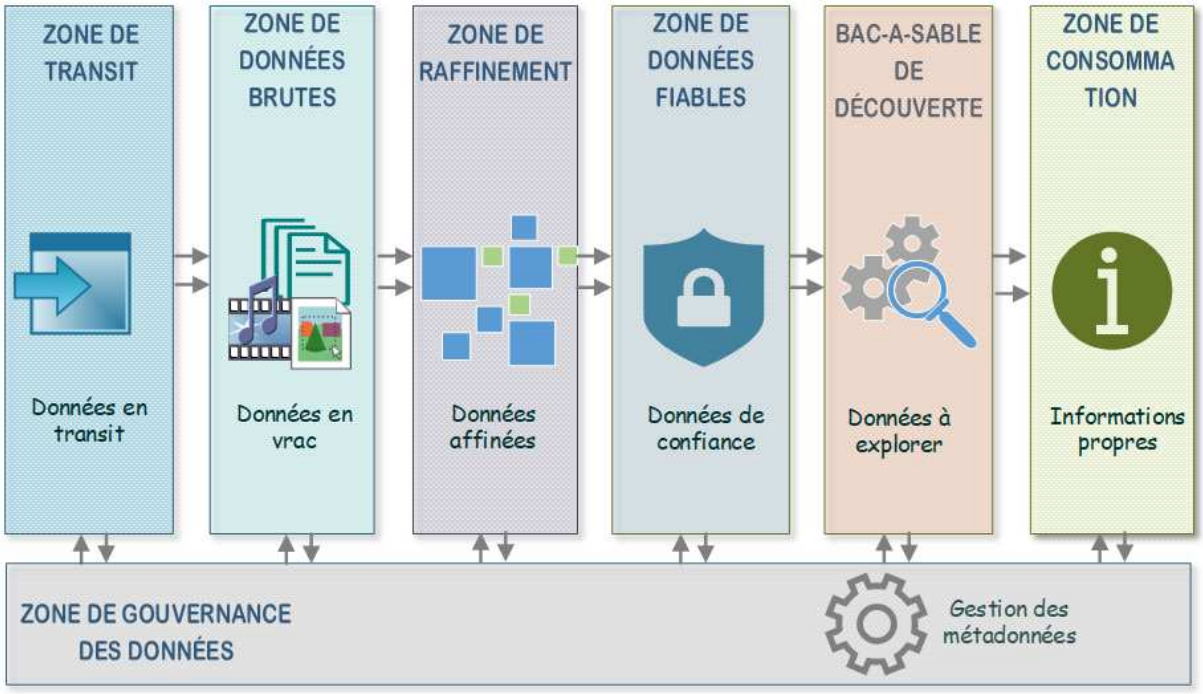


Figure 4.3 – Architecture en zone de Zaloni (LaPlante & Sharma, 2016)

Architectures hybrides

Dans les architectures hybrides de lacs de données, les composants dépendent à la fois des fonctions classiques d'un lac (stockage, traitements, analyses) et du degré de raffinement des données.

Un premier exemple d'architecture hybride est celle de B. Inmon (2016), qui propose une organisation du lac en un ensemble de bassins de données, c'est-à-dire de subdivisions du lac gérant chacun un type de données spécifique. D'après les spécifications de B. Inmon (2016), chaque bassin de données est associé à un système de stockage, à un ensemble de traitements et de prétraitements, ainsi qu'à des analyses spécifiques. Plus concrètement, B. Inmon (2016) définit cinq bassins de données (Figure 4.4).

1. Le **bassin de données brutes** traite les données brutes nouvellement ingérées. Il s'agit en fait d'une zone de transit, puisque les données sont ensuite conditionnées et transférées dans l'un des bassins de données analogiques, applicatives ou textuelles.

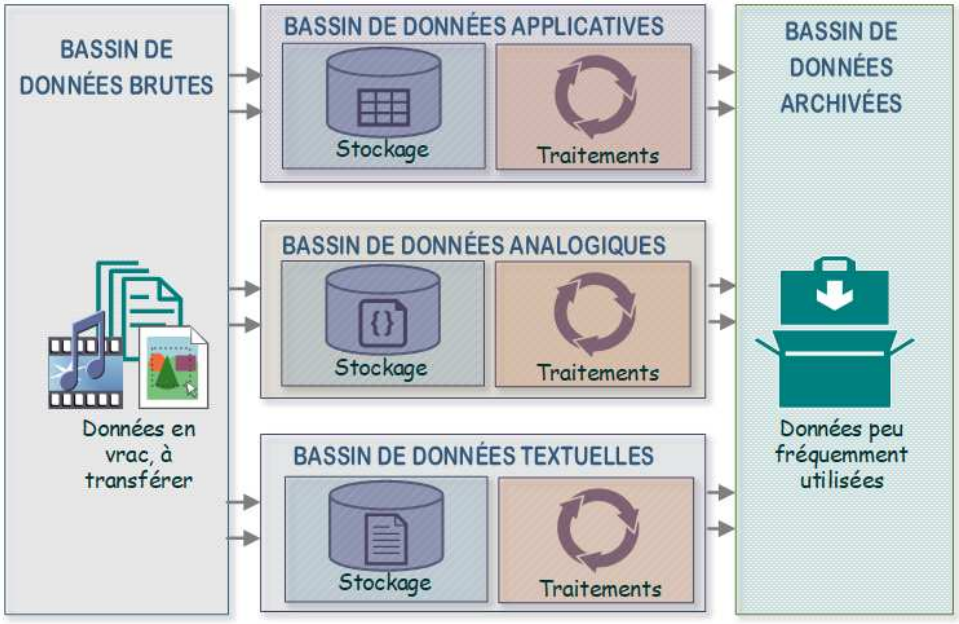


FIGURE 4.4 – Architecture en bassins données d'B. INMON (2016)

Contrairement aux autres bassins de données, le bassin de données brutes n'est associé à aucun système de métadonnées.

2. Le **bassin de données analogiques** prend en charge des données variées, caractérisées par une grande vélocité. Il s'agit généralement de données en flux provenant d'objets connectés.
3. Le **bassin de données applicatives** contient des données provenant d'un système OLTP. Ce sont généralement des données structurées, qui sont alors intégrées, transformées et préparées pour les analyses.
4. Le **bassin de données textuelles** gère les données textuelles, non structurées. À ce bassin est associé un processus de désambiguïsation textuelle qui prépare ces données aux analyses futures.
5. Le **bassin de données archivées** permet enfin de sauvegarder les données qui ne sont pas/plus activement utilisées, mais qui pourraient être de nouveau utiles. Ces données sont transférées depuis les bassins de données textuelles, applicatives ou analogiques.

Cette architecture peut être considérée comme hybride, car elle est à la fois basée sur la maturité des données et sur des fonctions. D'une part, l'architecture de B. INMON (2016) est basée sur la maturité des données car les données brutes sont gérées dans un composant spécial, à savoir le bassin de données brutes, tandis que les données raffinées sont gérées dans d'autres bassins. D'autre part, elle est aussi fonctionnelle car elle prévoit des composants de stockage et de traitement spécifiques répartis dans les bassins de données (Figure 4.4).

RAVAT et ZHAO (2019b) proposent également une architecture hybride, constituée de quatre composants de base, à l'image des architectures en zones (Figure 4.5) :

1. une zone de données brutes,

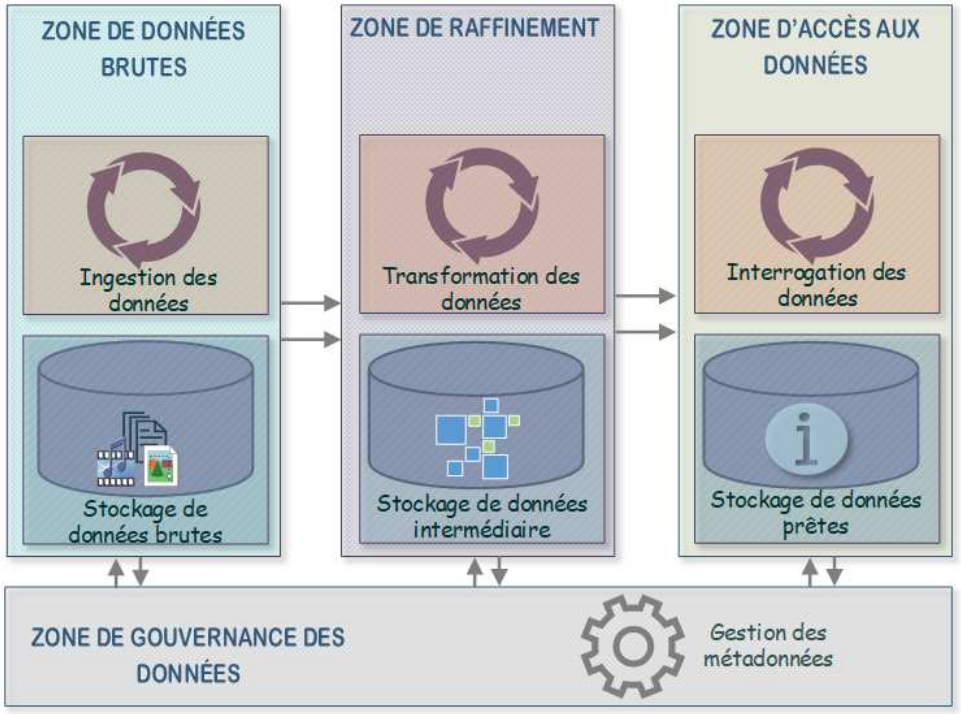


FIGURE 4.5 – Architecture hybride de RAVAT et ZHAO (2019b)

2. une zone de raffinement,
3. une zone de données exploitables,
4. une zone de gouvernance des données.

Dans chacune de ces zones, les fonctions de stockage et de traitement sont représentées par des sous-composants imbriqués.

Discussion Les architectures fonctionnelles ont l'avantage de mettre clairement en évidence les fonctions à implémenter lors de la construction d'un lac de données. Elles permettent ainsi d'identifier plus facilement les technologies requises. Les architectures basées sur la maturité des données permettent quant à elles de planifier et d'organiser le cycle de vie des données.

Cependant, chacune de ces deux approches a des limites, car ne considérant qu'une parmi plusieurs dimensions de l'architecture. Or, il est important de notre point de vue de prendre en compte à la fois les fonctions à implémenter et le cycle de vie des données lors de la conception architecturale d'un lac de données.

Par conséquent, nous préconisons l'adoption d'approches hybrides. Cependant, les architectures hybrides existantes peuvent encore être améliorées. En effet, dans l'approche de B. INMON (2016), les données brutes sont supprimées une fois qu'elles sont raffinées. Cela peut induire une perte de données, ce qui est contraire à l'esprit des lacs de données. De même, dans la proposition de RAVAT et ZHAO (2019b), l'accès aux données ne semble possible que pour les données raffinées.

Plus récemment, GIEBLER et al. (2021) ont proposé une plateforme permettant d'aboutir

à une architecture hybride de lac de données. Ce *framework* se base sur neuf aspects de la conception d'un lac de données :

1. l'infrastructure physique (par exemple, Hadoop) ;
2. le type de stockage (par exemple, NoSQL) ;
3. la nature des données (en flux ou en lots) ;
4. la modélisation des données ;
5. le cycle de vie des données (zones) ;
6. les traitements appliqués aux données ;
7. la gestion des métadonnées ;
8. la sécurité et la confidentialité des données ;
9. la gestion de la qualité des données.

La multiplicité des perspectives prises en compte dans ce *framework* architectural lui permet de surpasser les limites des architectures pré-existantes. Cependant, certaines des dimensions considérées par Giebler et al. (2021) nous semblent redondantes, ce qui induit un flou préjudiciable. Par exemple, l'infrastructure physique (dimension n° 1) et le type de stockage (dimension n° 2) sont hautement dépendants. De même, la modélisation des données (dimension n° 4) entre dans le cadre plus large de la gestion des métadonnées (dimension n° 7). Enfin, le cycle de vie des données (dimension n° 5) détermine la gestion de la sécurité des données (dimension n° 8) ainsi que la qualité des données (dimension n° 9). Par conséquent, une fusion de certaines dimensions de conception nous paraît nécessaire pour rendre cette plateforme plus compréhensible et donc plus applicable.

4.2.2 Nouvelles architectures des systèmes décisionnels

Les architectures décisionnelles traditionnelles comportent généralement des systèmes comme l'ODS, l'entrepôt et les magasins de données (Section 2.2.3). Dans ces architectures, les données passent des sources de données à l'entrepôt (et aux magasins de données) à travers le classique processus d'ETL. Mais le nouveau concept de lac de données a engendré une évolution de cette architecture classique. En effet, son arrivée dans l'architecture décisionnelle a nécessité de repenser le cycle de vie des données, ainsi que la distribution des rôles à travers les différents systèmes décisionnels. Ainsi, trois approches ont été proposées pour combiner le lac de données avec les systèmes décisionnels traditionnels et notamment les entrepôts de données.

Entrepôt de données en aval du lac de données

Cette approche vise à tirer le meilleur parti de chacun des entrepôts et des lacs. Les lacs de données permettent en effet de stocker plus facilement et à moindre coût de grandes quantités de données brutes. Ils sont donc exploités comme des zones de transit ou ODS (Fang, 2015 ; Russom, 2017). L'entrepôt de données vient alors en aval du lac de données. Il reçoit les données préalablement nettoyées et affinées dans le lac de données, via le classique processus d'ETL. Suivant cette approche, l'entrepôt de données est dédié à des analyses industrialisées mises en œuvre à l'aide des langages classiques d'analyse multidimensionnelle, c'est-à-dire SQL et MDX. Le lac de données est quant à lui utilisé pour des analyses plus avancées, mais ponctuelles, via des scripts (Figure 4.6).

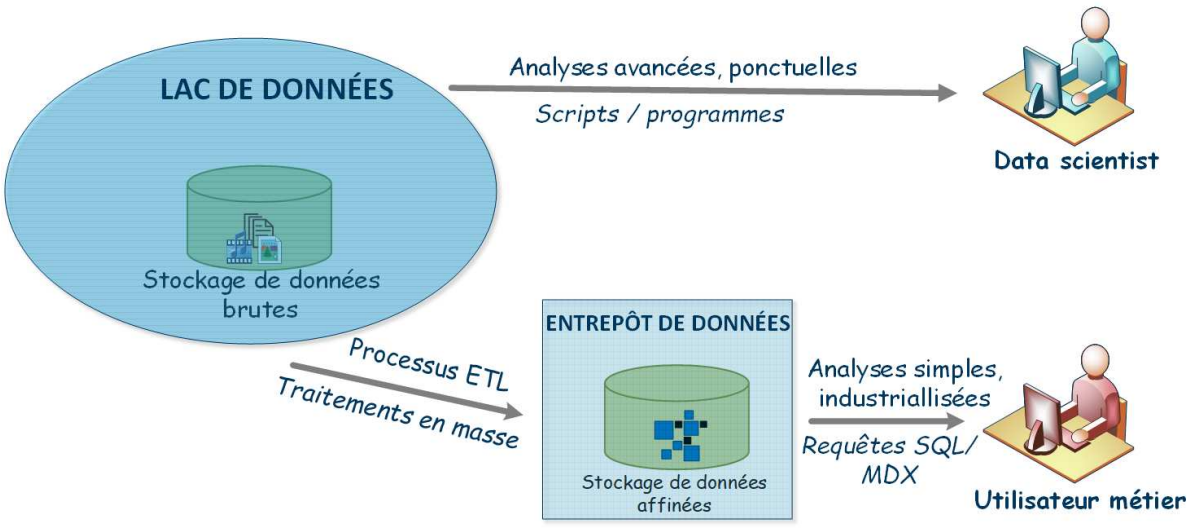


FIGURE 4.6 – Architecture « entrepôt en aval du lac »

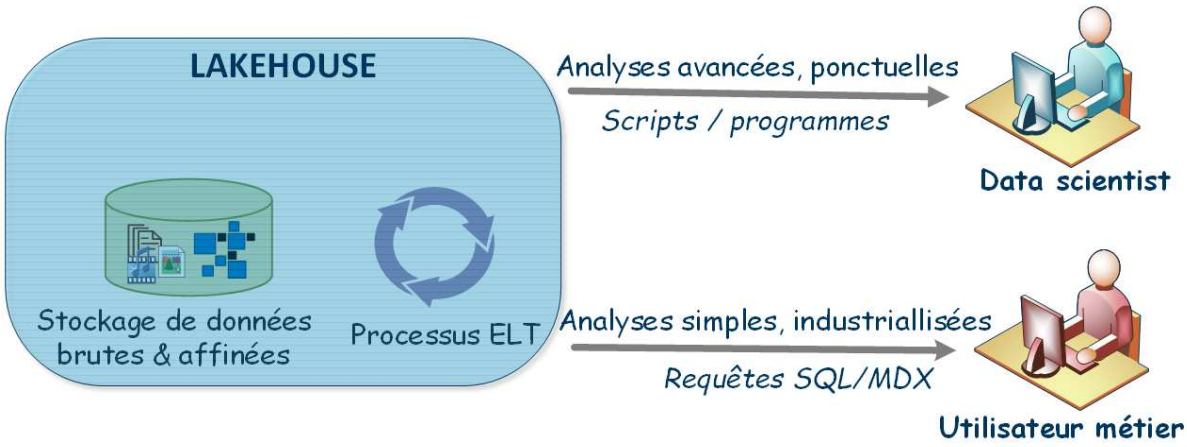


FIGURE 4.7 – Architecture « entrepôt fusionné au lac »

Entrepôt de données dans le lac de données

Comme détaillé dans la Section 4.2.1, B. INMON (2016) propose une architecture basée sur une subdivision du lac de données en bassins de données. Suivant ses spécifications, un de ces bassins de données, en l'occurrence le bassin des données applicatives, représente purement et simplement un entrepôt de données. Cette approche considère donc le lac de données comme une extension de l'entrepôt de données.

Entrepôt de données fusionné avec le lac de données

Une approche plus récente consiste à combiner un lac de données avec un entrepôt en un nouveau système dénommé « *lakehouse* » (ARMBRUST et al., 2021). Le *lakehouse* vise ainsi à proposer à la fois des fonctionnalités propres aux entrepôts de données (caractéristiques ACID, requêtes SQL) et aux lacs de données (versionnement des données, indexation,

stockage peu coûteux, etc.). Dans le *lakehouse*, toutes les analyses sont réalisées sans distinction depuis une interface unique (Figure 4.7).

Discussion

Chacune de ces trois approches architecturales vise à tirer le meilleur parti des bénéfices associés à chacun des deux systèmes décisionnels, entrepôt et lac. Les divergences se trouvent surtout dans leurs approches de mise en œuvre. Les deux premières approches (« entrepôt en aval du lac » et « entrepôt dans le lac ») sont les plus simples à implémenter, car ne faisant qu'étendre l'entrepôt de données. C'est pourquoi, elles ont été largement adoptées dès la naissance du concept de lac de données. Cependant, ces approches induisent un problème de silotage des données qui se manifeste par une séparation des données brutes et raffinées (dans la première approche) ou via une isolation des données structurées du reste des données (dans la deuxième approche).

La troisième approche s'affranchit de cette limitation en organisant les données à travers un système global, indépendamment de leurs types et de leurs niveaux de raffinement. Cela implique cependant une plus grande complexité de mise en œuvre. D'ailleurs, le concept de *lakehouse* demeure à ce jour plutôt théorique. Certes, la technologie des *delta lakes* (Chockalingam, 2019) propose les fonctionnalités nécessaires à de tels systèmes, mais ces solution restent encore relativement immatures et doivent encore être éprouvées.

4.3 Technologies et méthodes utilisées dans les lacs de données

La majorité des implémentations de lacs de données est basée sur la technologie Apache Hadoop [1]. Hadoop présente en effet l'avantage de fournir à la fois une solution de stockage et des outils de traitement des données. Toutefois, il existe des technologies alternatives à Hadoop. Dans cette section, nous présentons ces technologies alternatives, ainsi qu'une large panoplie de méthodes communément exploitées pour ingérer, stocker, traiter, contrôler et analyser les données dans un lac.

4.3.1 Ingestion des données

Les techniques et technologies d'ingestion des données servent à transférer physiquement les données dans le lac depuis les sources de données. Une première catégorie d'outils est constituée de logiciels qui collectent itérativement les données sous la forme d'un ensemble de tâches industrialisées. Plusieurs de ces outils sont fournis par la fondation Apache [2], parmi lesquels on peut citer Flink [3] et Samza [4] (qui sont des outils de traitement parallèle utilisés pour le transfert de données en flux), Flume [5] (qui est spécialisé dans la collecte, le traitement et le transfert de *logs*) et Kafka [6] qui propose un traitement en temps réel

1. https://hadoop.apache.org/
2. https://www.apache.org/
3. https://flink.apache.org/
4. http://samza.apache.org/
5. https://flume.apache.org/
6. https://kafka.apache.org/

tolérant aux pannes (John & Misra, 2017 ; Mathis, 2017).

Une deuxième catégorie de technologies d'ingestion de données est constituée d'outils et de protocoles réseaux couramment utilisés pour le transfert des données. Il s'agit notamment des protocoles *wget*, *rsync*, FTP et HTTP, qui peuvent être exploités par le gestionnaire du lac pour la création de scripts d'ingestion de données. Ces protocoles présentent l'avantage essentiel d'être très courants et facilement utilisables (Terrizzano et al., 2015). De même, plusieurs API sont proposées par les sources de données elles-mêmes pour l'extraction et le transfert des données. Par exemple, les plateformes CKAN[7] et Socrata[8] fournissent des API pour accéder à leurs catalogues de données ouvertes, ainsi qu'aux métadonnées associées (Terrizzano et al., 2015).

4.3.2 Stockage des données

Il existe une large panoplie de technologies et de techniques utilisées et utilisables pour le stockage des données dans les lacs. Nous les regroupons en six catégories.

HDFS

Les données sont très souvent stockées dans les lacs de données à l'aide d'un système de fichiers dont le plus emblématique est *Hadoop Distributed File System* (HDFS). Une enquête réalisée par Russom (2017) montrait d'ailleurs que ce type de stockage était utilisé (seul ou en conjonction avec d'autres) dans environ 75 % des implémentations de lacs de données.

Cela est probablement imputable aux nombreux avantages offerts par cette technologie. En effet, HDFS propose un stockage distribué des données à travers plusieurs machines, ce qui lui permet de passer aisément à l'échelle et d'être tolérant aux pannes, c'est-à-dire que le système arrive à fonctionner quasi-normalement malgré l'indisponibilité éventuelle d'une ou plusieurs machines (John & Misra, 2017). HDFS présente également l'avantage de convenir au stockage en vrac de données brutes (Lepine, 2018). Il est donc particulièrement adapté au stockage des données non structurées, lesquelles constituent la majorité des données potentielles dans les lacs de données.

Cependant, HDFS convient peu aux fichiers de taille reduite (John & Misra, 2017). En effet, les fichiers sont stockés dans HDFS à travers des blocs de grande taille (64 Mo par défaut). Ainsi, un fichier de faible taille (quelques Mo) induit des coûts de stockage et d'accès disproportionnés. C'est pourquoi on lui préfère parfois des systèmes de fichiers classiques (Houle, 2017). De plus, HDFS seul n'est pas suffisant pour traiter tous les formats de données, en particulier les données structurées. Ainsi, il devrait idéalement être combiné avec des SGBD relationnels ou NoSQL.

7. https://ckan.org/
8. https://data.europa.eu/euodp/fr/node/6541

Systèmes de gestion de bases de données

Les SGBD relationnels comme MySQL [9], PostgreSQL [10] et Oracle [11] sont utilisés pour le stockage des données structurées (Beheshti et al., 2017 ; Khine & Wang, 2017). Cependant, ces outils de stockage conviennent peu aux données extrêmement volumineuses de par leurs propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité) qui limitent leurs capacités de passage à l'échelle (Krishnan, 2013). De plus, ils sont peu adaptés dans un contexte de données semi-structurées et surtout non structurées. En effet, les données sont communément stockées dans les SGBD relationnels sur la base d'un schéma prédéfini. Or, les lacs de données exigent une flexibilité dans le schéma des données.

C'est pourquoi les SGBD non relationnels sont communément utilisés à la place (Beheshti et al., 2017 ; Khine & Wang, 2017 ; Pathirana, 2015). Ces systèmes, encore appelés NoSQL (*Not Only SQL*) assurent le passage à l'échelle en distribuant et répliquant les données à travers plusieurs machines. Cependant, ces systèmes distribués sont soumis à des limites exprimées par le théorème de Brewer (2000). Ce théorème, également connu sous l'acronyme CAP (*Consistency, Availability, Partition-tolerance*), montre en effet que seulement deux des trois caractéristiques de cohérence (*consistency*), de disponibilité (*availability*) et de tolérance aux pannes (*partition-tolerance*) peuvent être assurées dans de tels systèmes. La cohérence exprime le fait que toutes la machines sur lesquelles les données sont distribuées contiennent exactement les même données à chaque instant. La disponibilité consiste à garantir une réponse à chaque requête. La tolérance aux pannes représente le fait que le système continue de fonctionner malgré l'indisponibilité d'une partie des machines sur lesquelles les données sont distribuées.

À partir du théorème CAP, les SGBD peuvent être catégorisés en trois types (Krishnan, 2013). Une première catégorie de SGBD assurent la cohérence et la tolérance aux pannes (CP de CAP). Une deuxième catégorie opte pour la disponibilité et la tolérance aux pannes (AP de CAP). Enfin, la troisième catégorie assure la cohérence et la disponibilité (CA de CAP). C'est typiquement le cas des SGBD relationnels (Figure 4.8).

Plus spécifiquement, les systèmes NoSQL peuvent par ailleurs être classifiés en quatre types, suivant l'approche de structuration des données (John & Misra, 2017 ; Pathirana, 2015). Chacune de ces catégories de systèmes répond alors à un besoin de stockage spécifique.

1. Les **systèmes clés-valeurs** sont essentiellement des tables de hashage qui associent une clé aux données à stocker. Les couples « clés-valeurs » sont imbriqués de sorte à pouvoir retrouver à la fois des données atomiques et complexes, et ce de façon rapide.
   Exemples : Redis [12], SimpleDB [13] et Memcached [14].
2. Les **systèmes orientés documents** peuvent être considérés comme une spécialisation des systèmes clés-valeurs particulièrement adaptés pour le stockage de données

9. https://www.mysql.com/
10. https://www.postgresql.org/
11. https://www.oracle.com/fr/database/technologies/
12. https://redis.io/
13. https://dbdb.io/db/simpledb
14. https://memcached.org/

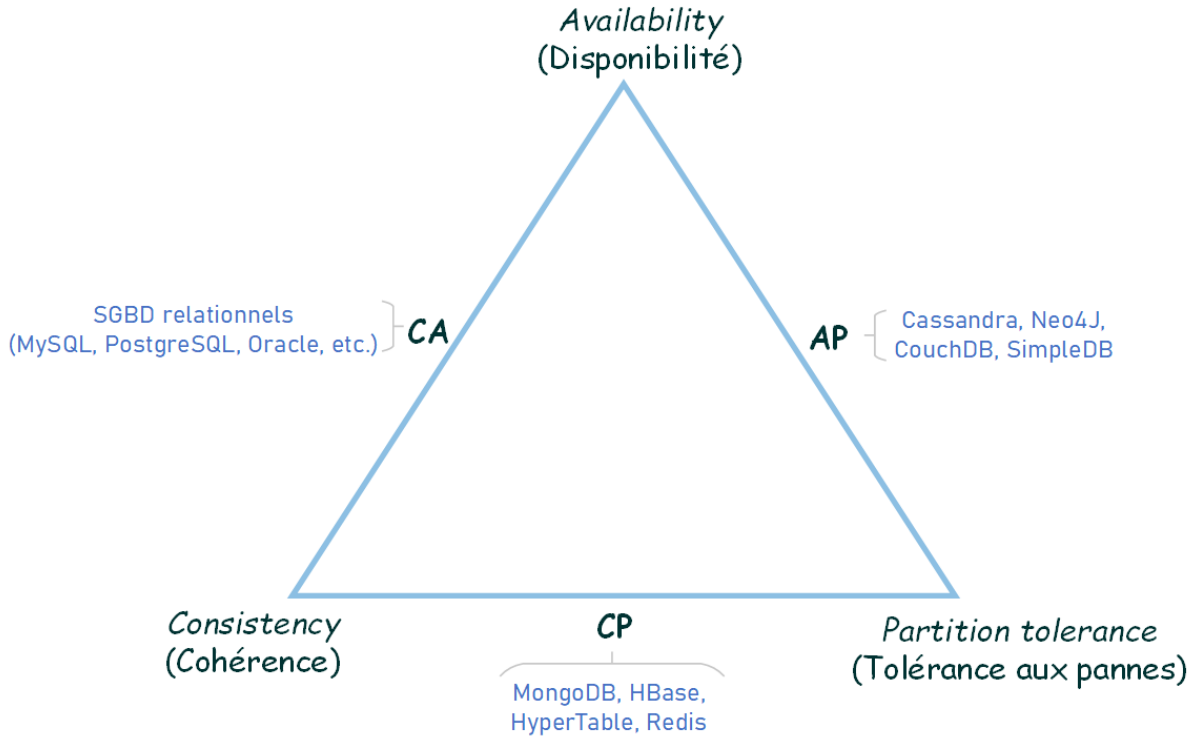


FIGURE 4.8 – Catégorisation des SGBD suivant le théorème CAP

semi-structurées à schéma hétérogène ou évolutif.
Exemples : CouchDB [15], MongoDB [16] et BaseX [17].

3. Les **systèmes orientés colonnes** stockent des tables relationnelles sous la forme d'un ensemble de colonnes, contrairement au stockage par n-uplets classiquement utilisé. Ils facilitent ainsi les opérations d'agrégation et permettent des requêtes SQL.
   Exemples : Cassandra [18], HBase [19] et Hypertable [20].
4. Les **systèmes orientés graphes** se focalisent sur le stockage de relations. Ils adoptent donc une structure en graphes, qui sied particulièrement au stockage de données provenant des réseaux sociaux.
   Exemples : Neo4j [21], AllegroGraph [22] et HyperGraphDB [23].

15. https://couchdb.apache.org/
16. https://www.mongodb.com/
17. https://basex.org/
18. https://cassandra.apache.org/
19. https://hbase.apache.org/
20. https://hypertable.org/
21. https://neo4j.com/
22. https://allegrograph.com/
23. http://www.hypergraphdb.org/

Systèmes multi-paradigmes

Face à l'hétérogénéité des données dans les lacs de données, des systèmes multi-paradigmes ont été proposés (Nogueira et al., 2018). Ces systèmes permettent de stocker différents types de données en combinant plusieurs technologies de stockage. Ils peuvent être catégorisés en quatre types (Leclercq & Savonnet, 2018 ; Tan et al., 2017).

1. Les **bases de données fédérées** consistent en un ensemble de systèmes de stockage homogènes (plusieurs SGBD relationnels, par exemple) associés à une unique interface d'interrogation. Le système Multibase (Smith et al., 1981) est un exemple de base de données fédérée.
2. Les **systèmes polyglottes** sont quant à eux constitués d'un ensemble de systèmes de stockage homogènes associés à de multiples interfaces d'interrogation. Le framework Spark SQL (Armbrust et al., 2015) permet ainsi d'interroger des tables relationnelles à travers un langage procédural ou via des requêtes SQL classiques.
3. Les **multistores** associent des systèmes de stockage hétérogènes à une interface d'interrogation unique. C'est le cas du système Polybase (DeWitt et al., 2013), qui permet d'interroger à la fois des données stockées dans un entrepôt de données et dans HDFS via le langage SQL.
4. Les **polystores** consistent en des ensembles de systèmes de stockage hétérogènes, associés à plusieurs interfaces d'interrogation. Ils peuvent être vus comme une association de systèmes polyglotes et de multistores. Apache Drill[24] peut être considéré comme un exemple de polystore (Tan et al., 2017).

Le Tableau 4.1 résume les différences entre les quatre catégories de systèmes multi-paradigmes.

Table 4.1 – Catégories de systèmes de stockage multi-paradigmes

Fonctionnalité → **Système ↓**	**Système de stockage**	**Interface(s) d'interrogation**
Bases de données fédérées	Homogène	Unique
Systèmes polyglotes	Homogène	Multiples
Multistores	Hétérogène	Unique
Polystores	Hétérogène	Multiples

Stockage orienté objet

Le stockage orienté objet consiste à encapsuler les données brutes avec des métadonnées dans une même entité (Tiao, 2018). Ce mode de stockage est souvent utilisé dans les systèmes *cloud*, en guise d'alternative au stockage en blocs utilisé par HDFS (Mathis, 2017). Le stockage orienté objet est généralement mis en œuvre à l'aide des standards

24. https://drill.apache.org/

de fichiers auto-descriptifs tels que JSON[25], ORC[26] et AVRO[27] (Beheshti et al., 2017 ; Mathis, 2017). Il est donc particulièrement adapté à des interaction via des API, du fait de ces standards (Mathis, 2017). Le stockage orienté objet présente également l'avantage d'être peu coûteux, plus précisément trois à cinq fois moins coûteux que le stockage sous HDFS (Tiao, 2018).

Standards sémantiques du W3C

Ce mode de stockage est généralement utilisé pour prendre en charge des ressources sémantiques telles que les ontologies, les vocabulaires et les thésaurus. Il est également utilisé pour stocker des relations entre les données (Farid et al., 2016). Dans la littérature, les normes W3C[28] les plus citées sont les formats OWL (*Web Ontology Language*) et le standard RDF (*Resource Description Framework*) (Farid et al., 2016 ; Laskowski, 2016 ; Pathirana, 2015 ; Stein & Morrison, 2014). Cela est probablement imputable au puissant langage de requête SPARQL qui fonctionne sur les données RDF et permet ainsi une interrogation aisée des données.

Indexation

Pour faciliter l'interrogation des données non structurées et semi-structurées, une solution couramment utilisée consiste à les stocker dans un index (Beheshti et al., 2017 ; Haste, 2017). Un index regroupe les données de sorte à retrouver facilement et rapidement des données partageant des caractéristiques similaires (John & Misra, 2017). Bien que le stockage en index soit plus répandu pour des données textuelles, il reste tout de même pertinent pour les autres données d'un lac. L'index peut ainsi servir à rechercher par mots clés à travers un ensemble de données structurées. Il sert également à la recherche inversée, à l'image du service Google Images[29] qui permet de retrouver une image à partir d'un extrait. Des exemples d'outils d'indexation des données sont Attivio[30], Apache Solr[31], Elasticsearch[32] et Sinequa[33].

4.3.3 Interrogation et traitement des données

Interrogation

Les modes d'interrogation des données contenues dans les lacs dépendent fortement des technologies de stockage utilisées. Ainsi, dans le cas du stockage en SGBD, des langage de requêtes classiques comme le SQL (pour les SGBD relationnels), JSONiq[34] (pour MongoDB), XQuery[35] (pour les bases de données XML) ou SPARQL (pour les ressources

25. https://www.json.org/
26. https://orc.apache.org/
27. https://avro.apache.org/
28. https://www.w3.org/
29. https://www.google.fr/imghp
30. https://www.attivio.com/
31. https://solr.apache.org/
32. https://www.elastic.co/
33. https://www.sinequa.com/
34. https://www.jsoniq.org/
35. https://www.w3.org/XML/Query/

sémantiques) sont utilisés (Farid et al., 2016 ; Hai et al., 2016 ; Laskowski, 2016 ; Russom, 2017).

Cependant, ces langages ne permettent pas d'interroger simultanément plusieurs systèmes de stockage, comme c'est le cas dans des lacs de données implémentés via des systèmes de stockage multi-paradigmes (Section 4.3.2). Dans de tels cas, des techniques et technologies de réécriture de requêtes associées à ces systèmes multi-paradigmes sont utilisées. Ainsi, les *frameworks* Spark SQL (Armbrust et al., 2015) et SQL++ (Ong et al., 2014) permettent d'interroger à la fois des données de SGBD relationnels et des données semi-structurées au format JSON. Le *framework* SQRE (*Scalable Query Rewriting Engine*), proposé par Hai et al. (2018), va plus loin en prenant en charge des bases de données orientées graphes (en plus des bases de données relationnelles et orientées documents). Le langage de requêtes proposé par le système CloudMdsQL (Kolev et al., 2016) permet également d'interroger simultanément de multiples SGBD relationnels et non relationnels.

La fondation Apache propose également un ensemble d'outils dédiés à l'interrogation de systèmes de stockage hétérogènes. Apache Phoenix[36] permet ainsi de convertir automatiquement des requêtes SQL au langage natif de plusieurs systèmes NoSQL. Apache Drill[37] sert à réaliser des jointures entre des données stockées dans des SGBD distincts. Apache Pig[38] est utilisé pour l'extraction de données stockées dans HDFS (John & Misra, 2017).

Enfin, certaines implémentations de lacs de données proposent une interface sous forme d'API à travers laquelle les utilisateurs peuvent interagir avec les données du lac. C'est le cas des lacs de données CoreDB et CoreKG, dans lesquels une API REST permet aux utilisateurs d'ajouter, retrouver, modifier ou supprimer des données (Beheshti et al., 2017 ; Beheshti et al., 2018).

Traitements

Le traitement des données d'un lac est généralement réalisé à l'aide de scripts personnalisés, qui exploitent le paradigme Map-Reduce (John & Misra, 2017 ; Khine & Wang, 2017), un paradigme de traitement parallèle des données fourni avec Apache Hadoop. Son approche de distribution des tâches sur plusieurs machines permet un passage à l'échelle des traitements (Tiao, 2018). Si Map-Reduce est particulièrement efficace pour le traitement de données par lots, il est toutefois limité pour le traitement de données en flux car il stocke les données sur disque lors des étapes intermédiaires. C'est pourquoi des approches alternatives comme Apache Spark, Apache Flink et Apache Storm[39] lui sont souvent préférées. Ces outils, qui fonctionnent exclusivement en mémoire vive, permettent ainsi des traitements en temps réel. Les *frameworks* Map-Reduce et Spark sont parfois utilisés simultanément, avec Map-Reduce dédié au traitement de données volumineuses en lots et Spark pour les données véloces en flux (John & Misra, 2017 ; Suriarachchi & Plale, 2016). C'est justement cette approche qui est appliquée dans l'architecture Lambda (Section 4.2.1).

36. https://phoenix.apache.org/
37. https://drill.apache.org/
38. https://pig.apache.org/
39. https://storm.apache.org/

4.3.4 Surveillance et sécurisation des données

Les lacs de données sont généralement implémentés sur des systèmes de stockage et de traitement distribués (HDFS, SGBD NoSQL, Map-Reduce, Spark, etc.). Pour un fonctionnement adéquat et efficient de ces systèmes, il est nécessaire de coordonner et d'optimiser l'utilisation des ressources par les différentes tâches. Dans Hadoop, ce travail est réalisé par l'outil YARN (*Yet Another Resource Negociator*), qui est chargé d'ordonnancer les tâches à exécuter et d'allouer les ressources, de sorte à assurer une exécution effective et optimale des applications. YARN est parfois utilisé en conjonction avec Apache Ambari [40], qui contrôle plus particulièrement les services propres au système Hadoop. Dans Hadoop ou indépendemment de Hadoop, Oozie [41] et Zookeeper [42] peuvent aussi servir à synchroniser et surveiller l'exécution de tâches.

Cependant, aucun des outils susmentionnés ne garantit la fiabilité des scripts en cours d'exécution. Ainsi, des scripts malveillants pourraient endommager les données sensibles d'un lac. Pour y remédier, le couple Yara [43] et Cuckoo [44] permet de traquer les *malwares*. Plus concrètement, le *framework* Yara fournit des descriptions de logiciels malveillants basées sur des modèles textuels ou binaires, qui peuvent alors être exploités par Cuckoo pour détecter automatiquement des scripts potentiellement malveillants (Cha et al., 2018).

4.4 Systèmes de lacs de données prêts à l'usage

Dans la section précédente, nous avons présenté les techniques et technologies utilisables pour mettre en œuvre les différentes fonctions d'un lac de données que sont l'ingestion des données (Section 4.3.1), le stockage (Section 4.3.2), l'accès aux données (Section 4.3.3) et la surveillance de l'infrastructure (Section 4.3.4). L'intégration de ces technologies en un système cohérent est parfois compliquée par des incompatibilités entre outils logiciels. C'est pourquoi certaines plateformes proposent, sous la forme d'un *package* unique, un ensemble de technologies mettant en œuvre ces fonctionnalités. Une partie de ces plateformes sont fournies sous la forme de systèmes à implémenter en local (Section 4.4.1), c'est-à-dire au sein des entreprises, et d'autres comme des services hébergés à distance, dans le *cloud* (Section 4.4.2).

4.4.1 Systèmes en local

Il s'agit ici de distributions Hadoop qui offrent un *package* complet des outils nécessaires au stockage et au traitement des données au sein d'une entreprise. Elles sont principalement basées sur des technologies Apache comme HDFS, YARN, Spark, Map-Reduce, etc., ou des équivalents. Ces distributions ont l'avantage de faciliter l'installation des différents composants et d'assurer leurs compatibilités mutuelles (Parageaud, 2013). De plus,

40. https://ambari.apache.org/
41. https://oozie.apache.org/
42. https://zookeeper.apache.org/
43. http://virustotal.github.io/yara/
44. https://cuckoosandbox.org/

elles sont caractérisées par des communautés fortes et dynamiques qui soutiennent les utilisateurs.

Les principales distributions Hadoop sont *Cloudera Distribution for Hadoop* (CDH)[45], *Hortonworks Data Platform* (HDP)[46] et *MAPR Data Platform*[47] (John & Misra, 2017). En 2019, les distributions CDH et HDP ont fusionné pour donner naissance à *Cloudera Data Platform* (CDP)[48]. L'existence de CDH et HDP se poursuit toutefois en sus de CDP jusqu'en 2022 (Rosencrance, 2019).

4.4.2 Services infonuagiques

Les services *cloud* de lacs de données comprennent un ensemble de technologies pour le stockage des données et l'analyse des *big data*, elles aussi hébergées dans le *cloud*. Ces services sont entièrement gérés par les fournisseurs. Comme les distributions Hadoop en local, ces services sont souvent basés sur Hadoop, mais comprennent également des technologies propriétaires. Par exemple, la plateforme Microsoft Azure fournit un système de fichiers appelé Cosmos en plus de HDFS (Ramakrishnan et al., 2017 ; Sirosh, 2016).

Ces services *cloud* offrent un avantage de flexibilité aux entreprises. En effet, la tarification du stockage et des traitements réalisés sur ces plateformes est continuellement ajustable à l'usage réel. Cela permet aux entreprises d'étendre ou de réduire à volonté les ressources à leur dispositions, et donc de mieux maîtriser les coûts engendrés (Sirosh, 2016).

Quelques exemples de plateformes proposant des services de lacs de données *cloud* sont *Amazon Web Services* (AWS)[49], *Google Cloud Platform*[50], *Microsoft Azure*[51] et *Alibaba Cloud Platform*[52].

4.5 Conclusion

Dans ce chapitre, nous avons étudié les approches opérationnelles de conception d'un lac de données à travers les architectures, d'une part, et les techniques et technologies utilisées, d'autre part. Une partie de ce travail est inclus dans l'article de journal Sawadogo, P. N. & Darmont, J. (2021). On data lake architectures and metadata management. *Journal of Intelligent Information Systems*, *56*(1), 97-120. https://doi.org/10.1007/s10844-020-00608-7.

Plus concrètement, nous avons proposé dans ce chapitre une typologie des approches de structuration interne des lacs de données, qui distingue trois catégories. Les architectures fonctionnelles définissent les composants du lac en suivant des fonctions de base (par exemple, couche de stockage, couche de traitement, couche d'accès, etc.). Les architectures en zones subdivisent le lac de données en fonction du niveau de raffinement des

45. https://www.cloudera.com/products/open-source/apache-hadoop/key-cdh-components.html
46. https://fr.cloudera.com/products/hdp.html
47. https://mapr.com
48. https://fr.cloudera.com/products/cloudera-data-platform.html
49. https://aws.amazon.com/
50. https://cloud.google.com/
51. https://azure.microsoft.com/
52. https://eu.alibabacloud.com/

données. Enfin, les architectures hybrides adoptent à la fois une vision fonctionnelle et en zones.

Au delà de ces architectures « internes », le concept de lac de données a engendré une restructuration des systèmes d'information décisionnels. En effet, ces systèmes étant traditionnellement centrés sur l'entrepôt de données, l'introduction du nouveau concept de lac de données a nécessité de redéfinir une distribution des rôles entre les deux concepts. Nous avons ainsi identifié trois approches proposées dans la littérature pour combiner les lacs et les entrepôts de données. Dans la première approche, le lac est utilisé comme un ODS, en amont de l'entrepôt de données. La deuxième approche, proposée par B. Inmon (2016), considère l'entrepôt de données comme une composante du lac, dédiée aux données structurées. La troisième approche, la plus récente, propose une fusion des deux concepts pour former un *lakehouse*. Cette dernière approche a l'avantage de mieux tirer parti des avantages des entrepôts et des lac de données. Elle reste cependant relativement immature et doit encore être éprouvée.

Du point de vue des technologies et techniques associées aux lacs de données, nous avons proposé un ensemble d'outils pour l'ingestion, le stockage, le traitement et la surveillance des données et de l'infrastructure. Nous avons également identifié un ensemble de plateformes « clés en mains », permettant d'implémenter l'ensemble des fonctions d'un lac de données en local ou dans le *cloud*. La figure 4.9 fait la synthèse de ces technologies.

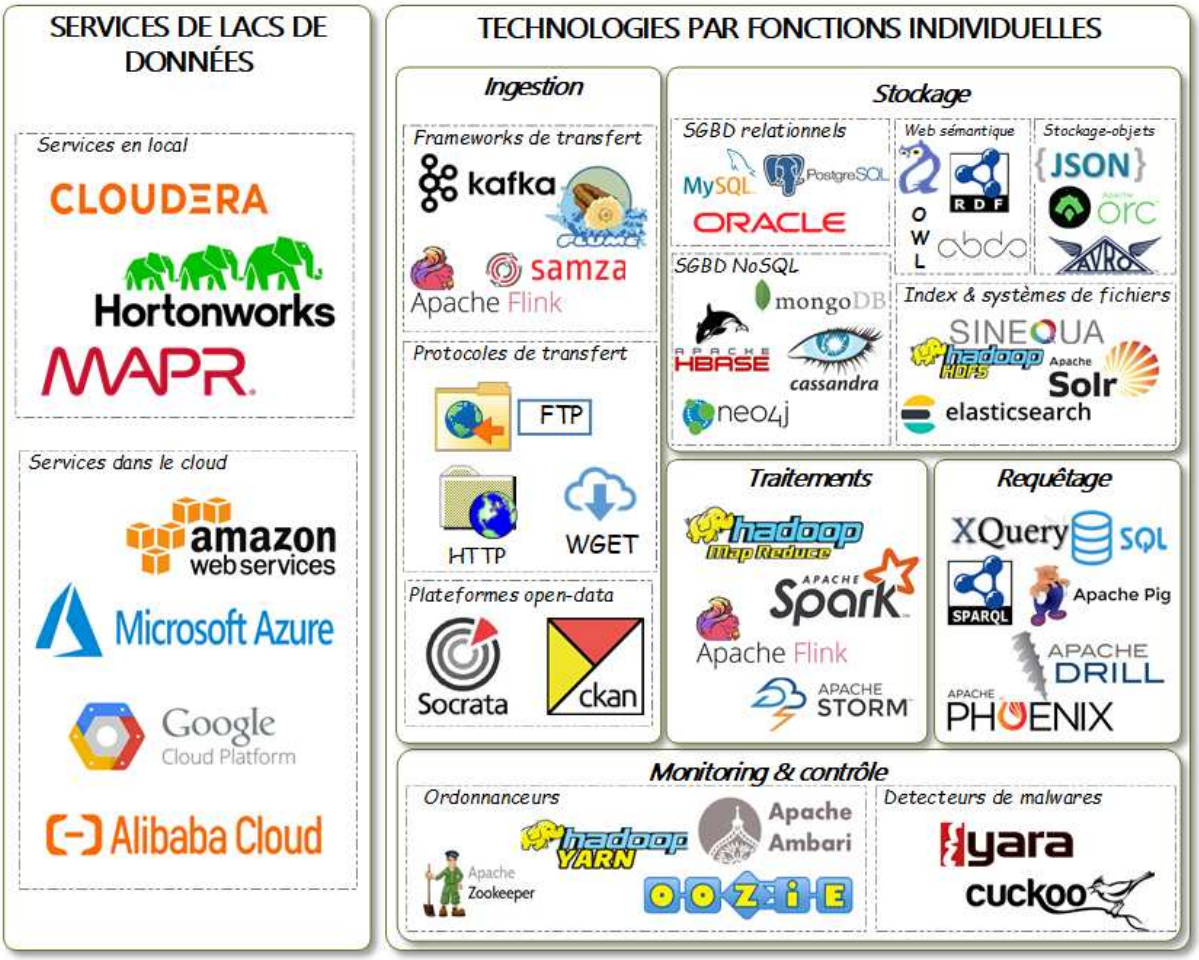


FIGURE 4.9 – Technologies associées aux lacs de données

Deuxième partie

Contributions sur la modélisation et l'organisation des métadonnées

Chapitre 5

Modèle de métadonnées MEDAL

Sommaire

5.1 Introduction

La conception d'un lac de données passe par la mise en place d'un système de métadonnées, qui à son tour nécessite un modèle de métadonnées pour sa mise en œuvre. Un modèle de métadonnées propose en effet un formalisme permettant de décrire de façon conceptuelle les relations entre les données et les métadonnées dans le lac. Il sert également de guide aux concepteurs du lac lors de l'identification des métadonnées adaptées à leur cas d'usage.

En vue de nous guider dans l'implémentation de notre lac de données textuelles et tabulaires (Section 1.2.1), nous introduisons dans ce chapitre un nouveau modèle de métadonnées. Nous avons proposé ce modèle, intitulé MEDAL (*MEtadata model for Data Lakes*), après les modèles GEMMS (Quix et al., 2016), Ground (Hellerstein et al., 2017) et de Diamantini et al. (2018) ; mais avant le modèle HANDLE (Eichler et al., 2020) et de façon quasi-concomitante au modèle de Ravat et Zhao (2019b). Au delà de notre cas d'usage, nous avons proposé MEDAL pour remédier aux limites des modèles pré-existants (Section 3.4).

À travers MEDAL, nous introduisons une nouvelle typologie des métadonnées dans les lacs de données, basée sur la façon dont elles sont reliées aux données brutes. Nous introduisons également une méthode d'organisation des métadonnées en exploitant la théorie des graphes. Enfin, nous proposons une méthode d'évaluation des systèmes et modèles de métadonnées en nous servant d'un ensemble de fonctionnalités usuelles dans les systèmes de métadonnées. Cette méthode nous permet d'évaluer MEDAL comparativement aux modèles et systèmes de métadonnées existants dans la littérature sur les lacs de données.

Nous avons conçu le modèle MEDAL en collaboration avec Étienne Scholly, également doctorant. La présentation de MEDAL a été publiée sous deux aspects à travers les articles Sawadogo, P. N., Scholly, E., Favre, C., Ferey, É., Loudcher, S. & Darmont, J. (2019b). Metadata Systems for Data Lakes : Models and Features. *BI and Big Data Applications - ADBIS 2019 Short Papers and Workshop, Bled, Slovenia* et Scholly, E., Sawadogo, P., Favre, C., Ferey, E., Loudcher, S. & Darmont, J. (2019). Systèmes de métadonnées dans les lacs de données : modélisation et fonctionnalités. *15e journées EDA Business Intelligence & Big Data (EDA 2019)*, 77-92 .

5.2 Typologie structurelle des métadonnées

À travers cette typologie, nous proposons de catégoriser les métadonnées par rapport à la façon dont elles sont associées aux entités de données. Nous distinguons ainsi trois catégories de métadonnées, à savoir les métadonnées intra-entité de données, inter-entités et globales, schématisées dans la Figure 5.1.

5.2.1 Métadonnées intra-entité

Ces métadonnées sont des caractéristiques directement associées de façon individuelle à des entités de données d'un lac. Nous les considérons comme des métadonnées de premier

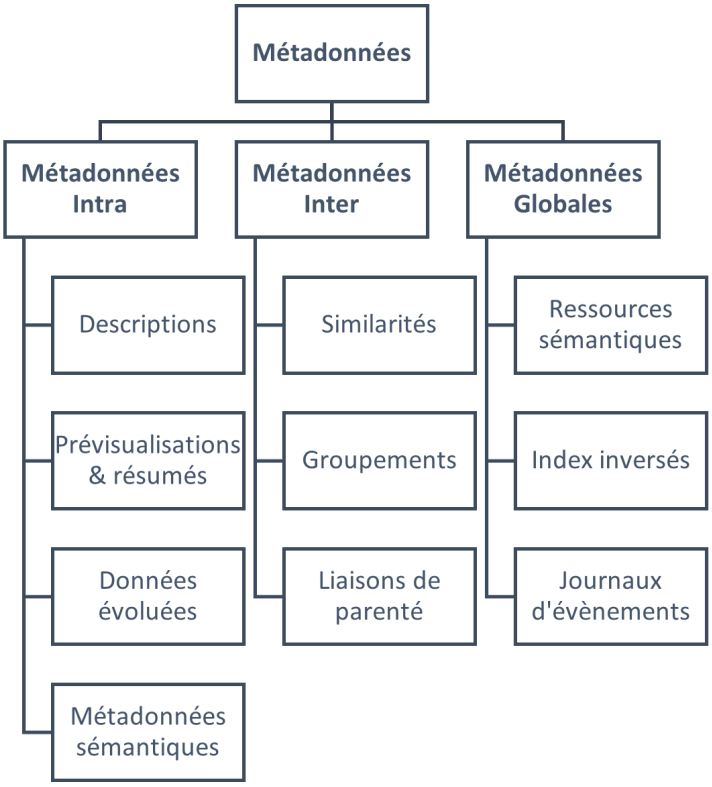


FIGURE 5.1 – Typologie structurelle de métadonnées

niveau et les classons en quatre sous-catégories.

1. Les **propriétés** sont des informations générales associées aux entités de données. Elles sont généralement obtenues à partir du système de fichiers, sous la forme de couples clés-valeurs. On peut citer par exemple le nom de l'entité de données, sa taille, son URL, son horodatage de création ou de modification, etc.

2. Les **prévisualisations et résumés** procurent un aperçu du contenu ou de la structure d'une entité de données. Il peut s'agir par exemple du schéma des données dans un contexte de données structurées ou semi-structurées, ou encore d'un nuage de mots dans le cas d'un document textuel.

3. Les **versions et représentations** sont les résultats de modifications réalisées sur une entité de données. Nous considérons en effet que lorsqu'une entité de données d' est générée dans le lac à partir d'une autre entité de données d, alors d' est une métadonnée de d (tout en restant une entité de données à part entière). Les versions sont ainsi issues d'opérations de mise à jour ou de modifications des données, alors que les représentations proviennent d'opérations de transformation des données. Cela peut consister, par exemple, à transformer un document textuel en un sac de mots, c'est-à-dire un vecteur, dans le but de permettre des traitements automatisés.

4. Les **métadonnées sémantiques** sont des annotations qui donnent du sens au contenu d'une entité de données. Il s'agit par exemple du titre, des descriptions ou encore de *tags*. Ces métadonnées sont soit fournies directement avec les entités de données, soit ajoutées manuellement par des utilisateurs métiers.

5.2.2 Métadonnées inter-entités

Les métadonnées inter-entités définissent des relations entre différentes entités de données dans le lac. Chaque instance de ces métadonnées n'est donc pas associée à une seule, mais à plusieurs entités de données simultanément. Nous les considérons comme des métadonnées de deuxième niveau, car elles procèdent des métadonnées intra-entité. Les métadonnées inter-entités peuvent être divisées en trois sous-catégories.

1. Les **groupements** permettent d'organiser les entités de données en collections. Chaque entité de données peut ainsi être associée à plusieurs groupes. Ces catégories sont généralement déduites des métadonnées intra-entité (*tags*, format des données, langage, auteur, etc.)
2. Les **relations de similarité** représentent une ressemblance ou une connexion entre des entités de données. Les dépendances fonctionnelles de type clé primaire-clé étrangère peuvent ainsi être traduites par des relations de similarité entre des entités de données tabulaires. C'est d'ailleurs à ce type de métadonnées que Maccioni et Torlone (2018) se réfèrent dans leur lac de données KAYAK à travers la notion de « joignabilité ».
3. Les **liaisons de parenté** servent à représenter la généalogie des données dans le cas d'une fusion. Elles complètent les versions et les représentations en conservant le processus ayant conduit à leur génération. Plus concrètement, lorsqu'une entité de données est générées à partir de données pré-existantes dans un lac, les entités initiales et l'entité fusionnée sont reliées par une relation de parenté. Les entités de données initiales sont alors considérées comme « parentes » et l'entité de données résultante comme « enfant ».

5.2.3 Métadonnées globales

Les métadonnées globales sont des structures qui servent principalement à faciliter et accélérer les traitements et analyses des données du lac. Elles ne sont pas directement associées à une quelconque entité de données du lac. Nous les considérons comme des métadonnées de troisième niveau. Nous en distinguons trois sous-catégories.

1. Les **ressources sémantiques** sont des bases de connaissances (ontologies, taxonomies, thésaurus, etc.) qui permettent d'améliorer les analyses et les traitements. Une ontologie peut par exemple servir à étendre automatiquement une requête par mots clés avec des termes équivalents.
2. Les **index inversés** sont des structures qui associent les entités de données du lac à des motifs (termes, phrases, extraits sonores) qu'ils contiennent. Elles permettent ainsi d'accélérer la recherche par mots clés et par motifs en général.
3. Les **journaux d'événements (ou *logs*)** tracent les interactions entre les utilisateurs et le lac de données. Il peut s'agir de simples opérations de connexion ou de déconnexion, ou encore des opérations plus complexes comme l'exécution d'un processus.

5.2.4 Comparaison à la typologie fonctionnelle de métadonnées

Comme nous l'avons relevé dans la Section 3.2.2, la typologie fonctionnelle d'Oram (2015) est insuffisamment précise et pourrait de ce fait induire des confusions lors de l'identification des métadonnées. La typologie structurelle que nous proposons a pour objectif d'être plus claire. Elle est aussi plus exhaustive quant à la panoplie de métadonnées utilisables dans un lac de données.

D'ailleurs, notre typologie inclue la plupart des métadonnées définies par Oram (2015). Les métadonnées métiers sont en effet comparables aux métadonnées sémantiques. Les métadonnées opérationnelles peuvent être considérées comme des journaux d'événements (*logs*) et les métadonnées techniques sont équivalentes aux métadonnées de prévisualisation. En somme, la typologie structurelle des métadonnées peut être considérée comme une extension, ainsi qu'une généralisation, de la classification fonctionnelle.

5.3 Modèle de métadonnées

Dans cette section, nous détaillons MEDAL, notre approche de modélisation des métadonnées. MEDAL se base sur la théorie des graphes pour représenter les métadonnées et les données, ainsi que les relations entre elles. Il se situe donc au niveau logique. Nous nous conformons cependant aux pratiques habituelles de modélisation en présentant comment MEDAL peut être perçu dans les trois niveaux de modélisation, c'est-à-dire aux niveaux conceptuel, logique et physique.

5.3.1 Modèle conceptuel de MEDAL

Le modèle MEDAL s'appuie sur un ensemble de concepts, intrinsèquement liés aux métadonnées identifiées dans la typologie structurelle (Section 5.2). Pour plus de clarté, nous présentons séparément les concepts utilisés pour chaque catégorie de métadonnées.

Représentation conceptuelle des métadonnées intra-entité

Conformément à notre typologie des métadonnées, nous considérons dans le lac de données trois types d'entités de données : les entités de données originelles, les représentations et les versions. Nous utilisons la qualification « originelles » pour représenter des données brutes, nouvellement ingérées dans le lac à partir de sources de données externes. Nous désignons par « représentations » les entités de données issues de la transformation d'une entité de données originelle. Enfin, le terme « version » représente une entité de données provenant de la modification (mise à jour) d'une entité de données originelle. Ces trois types d'entités de données, ainsi que les opérations de transformation et de modification, sont donc les concepts utilisés pour organiser conceptuellement les métadonnées intra-entité.

Définition 5.3.1. L'ensemble des entités de données e_i est noté $\mathcal{E} = \{e_i\}_{i \in \mathbb{N}^*}$.

Définition 5.3.2. L'ensemble des représentations d'une entité de données est noté $\mathcal{R}(e_i) = \{R_j(e_i)\}_{i \in \mathbb{N}^*, j \in \mathbb{N}^*}$, où $R_j : e_i \rightarrow e'_i$ est une opération de transformation et e'_i l'entité de

données résultant de la transformation.

Définition 5.3.3. L'ensemble des versions d'une entité de données est noté $\mathcal{V}(e_i) = \{V_k(e_i)\}_{i\in\mathbb{N}^*,k\in\mathbb{N}^*}$, où $V_k : e_i \rightarrow e_i''$ est une opération de modification et e_i'' l'entité de données résultant de la modification.

Définition 5.3.4. L'entité de données originelle associée à une entité de données e_i est notée $O(e_i)$.

Représentation conceptuelle des métadonnées inter-entités

De même, nous utilisons des concepts littéralement équivalents aux trois types de métadonnées définis dans cette catégorie. Ce sont les relations de similarité, les groupements et les liaisons de parenté. Les relations de similarité relient les entités de données deux à deux. Elles peuvent être orientées ou non. Les liaisons de parenté relient quant à elles plusieurs entités de données et sont toujours orientées. Pour représenter les groupements, nous utilisons le concept additionnel de « groupement ». Chaque groupement est constitué de groupes, auxquels les entités de données sont associées.

Définition 5.3.5. L'ensemble des relations de similarité s'écrit $\mathcal{S} = \{S_l\}_{l\in\mathbb{N}^*}$, avec $S_l = \{\mathcal{E} \times \mathcal{E}\}$.

Définition 5.3.6. L'ensemble des groupements est noté $\mathcal{G} = \{G_m\}_{m\in\mathbb{N}^*}$, avec $G_m = \{\Gamma_{mn}\}_{n\in\mathbb{N}^*}$, où $\Gamma_{mn} \subseteq \mathcal{E}$ est un groupe (une collection).

Définition 5.3.7. L'ensemble des relations de parenté est noté $\mathcal{P} = \{P_q\}_{q\in\mathbb{N}^*}$, avec $P_q = \{p_q \times e_q\}$, $p_q \subseteq \mathcal{E}$ l'ensemble des entités de données « parents » et e_q l'entité de données « enfant » générée.

Représentation conceptuelle des métadonnées globales

De notre avis, les métadonnées globales ne suscitent pas une véritable problématique de modélisation. Il existe en effet des méthodologies déjà éprouvées dans la littérature pour leur gestion. Les ressources sémantiques peuvent ainsi être représentées à l'aide du formalisme *Ontology-Based Data Access* (OBDA) (Bagozi et al., 2019) ou encore à l'aide du langage OWL. Les index inversés et les journaux d'événements sont quant à eux hautement dépendants des technologies utilisées pour leur implémentation. C'est pourquoi nous ne proposons pas de représentation particulière pour les métadonnées globales.

La Figure 5.2 représente les interactions entre les concepts de MEDAL.

5.3.2 Modèle logique de MEDAL

Dans cette section, nous proposons une traduction du modèle conceptuel de MEDAL (Section 5.3.1) au niveau logique. Nous organisons les métadonnées à l'aide d'éléments issus de la théorie des graphes.

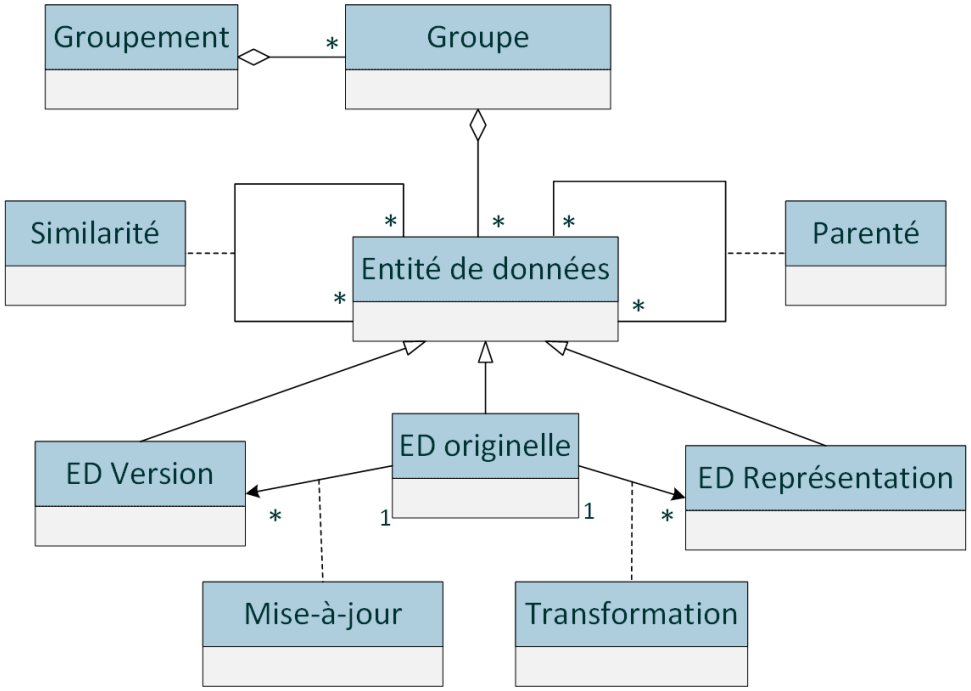


FIGURE 5.2 – Modèle conceptuel de MEDAL

Représentation logique des métadonnées intra-entité

Dans le modèle logique de MEDAL, nous utilisons un nœud pour représenter chaque entité de données (originelle, représentation ou version). Les nœuds traduisent non seulement ces éléments, mais aussi les métadonnées atomiques qui leur sont associées de façon individuelle. Des arêtes servent à exprimer les opérations de transformation (respectivement, de modification) entre les entités de données originelles et les représentations (respectivement, versions).

Nous utilisons un hyper-nœud pour rassembler chaque entité de données originelle avec l'ensemble de ses versions et représentations. L'hyper-nœud rassemble ainsi les métadonnées communes à toute la généalogie descendante d'une entité de données originelle. C'est par exemple le cas de la source des données ou encore de l'auteur. Nous considérons en effet que toutes les versions et représentations ont *a priori* la même origine que l'entité de données originelle dont elles sont issues.

La Figure 5.3 illustre la représentation des métadonnées intra-entité à travers l'exemple d'un CV. Le CV originel, une fois transformé en sac de mots, donne lieu à une représentation (en vert). Lorsqu'il est mis à jour, cela donne lieu à une nouvelle version (en orange). Sur le plan visuel, un hyper-nœud est représenté par un cercle avec un double contour. Les nœuds ont quant à eux des contours simples. Les métadonnées (titre, horodatages, descriptions, etc.) associées aux nœuds, hyper-nœuds et arêtes ne sont pas représentées, par souci de lisibilité.

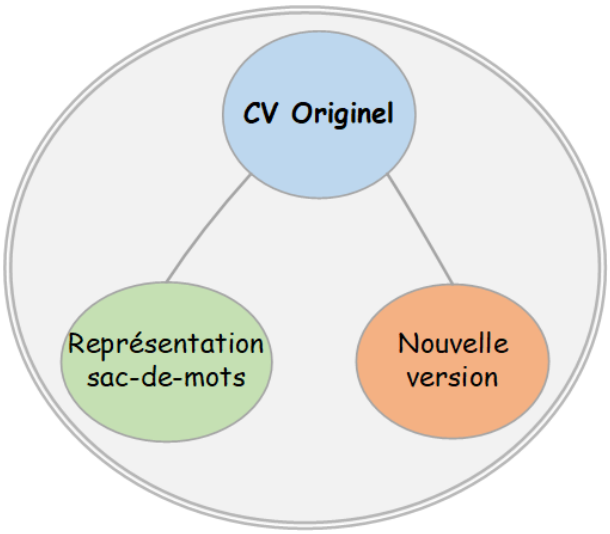


FIGURE 5.3 – Représentation logique des métadonnées intra-entité dans MEDAL

Représentation logique des métadonnées inter-entités

Pour organiser les métadonnées inter-entités, nous utilisons des arêtes et des hyper-arêtes. Une hyper-arête relie simultanément plusieurs nœuds. Elle nous sert notamment à matérialiser les groupements. Chaque groupe est ainsi représenté par une hyper-arête, qui relie un ensemble d'hyper-nœuds. Par exemple, en considérant un groupement sur la source des données, nous aurions une hyper-arête reliant les hyper-nœuds des entités de données issues des réseaux sociaux, une autre hyper-arête reliant ceux représentant des entités de données issues de plateformes de données ouvertes, etc. Les hyper-arêtes servent aussi à exprimer les liaisons de parenté, c'est-à-dire, les relations généalogiques entre des entités de données parents fusionnées et les entités de données résultantes. Dans ce cas, l'hyper-arête est orientée des nœuds parents vers le nœud représentant l'entité de données enfant.

Nous utilisons des arêtes simples pour représenter les relations de similarité entre des entités de données. Ces arêtes, qui relient les nœuds correspondants, peuvent être orientées ou non selon le type de relation. Par exemple, les relations de type clé primaire-clé étrangère sont orientées, tandis que les mesures de similarité comme la similarité cosinus (ALLAN et al., 2000) ne le sont pas.

La Figure 5.4 représente une hyper-arête orientée. Cet exemple présente le cas d'une nouvelle entité de données (fiches de produits) issue de la fusion de deux autres (liste de produits et données de ventes). De façon analogue, la Figure 5.5 représente deux hyper-nœuds traduisant un regroupement de CV en deux groupes selon la langue. Enfin, les relations de similarité sont représentées par des arêtes discontinues. La Figure 5.6 montre ainsi une mesure de la similarité cosinus appliquée entre des représentations en sac-de-mots de plusieurs documents textuels. Chaque hyper-arête est représentée visuellement à l'aide d'un cercle denté auquel sont associés les éléments liés. Dans le cas d'une hyper-arête orientée, une flèche sortante relie le cercle denté au nœud enfant.

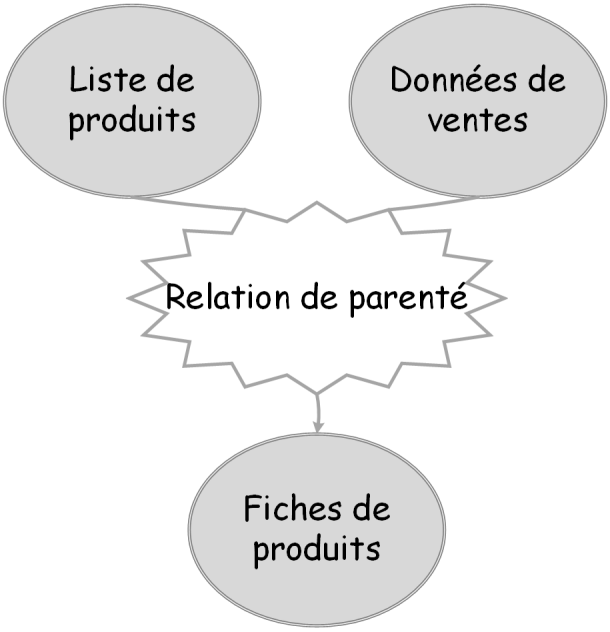


FIGURE 5.4 – Représentation logique des liaisons de parenté

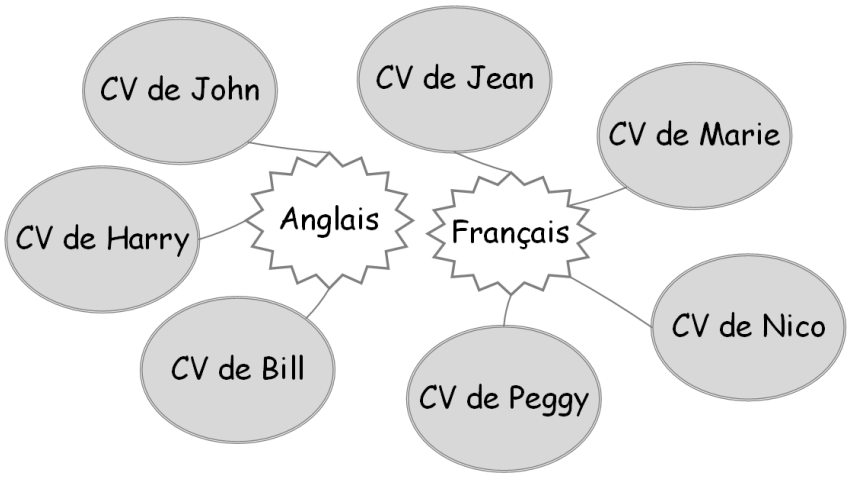


FIGURE 5.5 – Représentation logique des groupements dans MEDAL

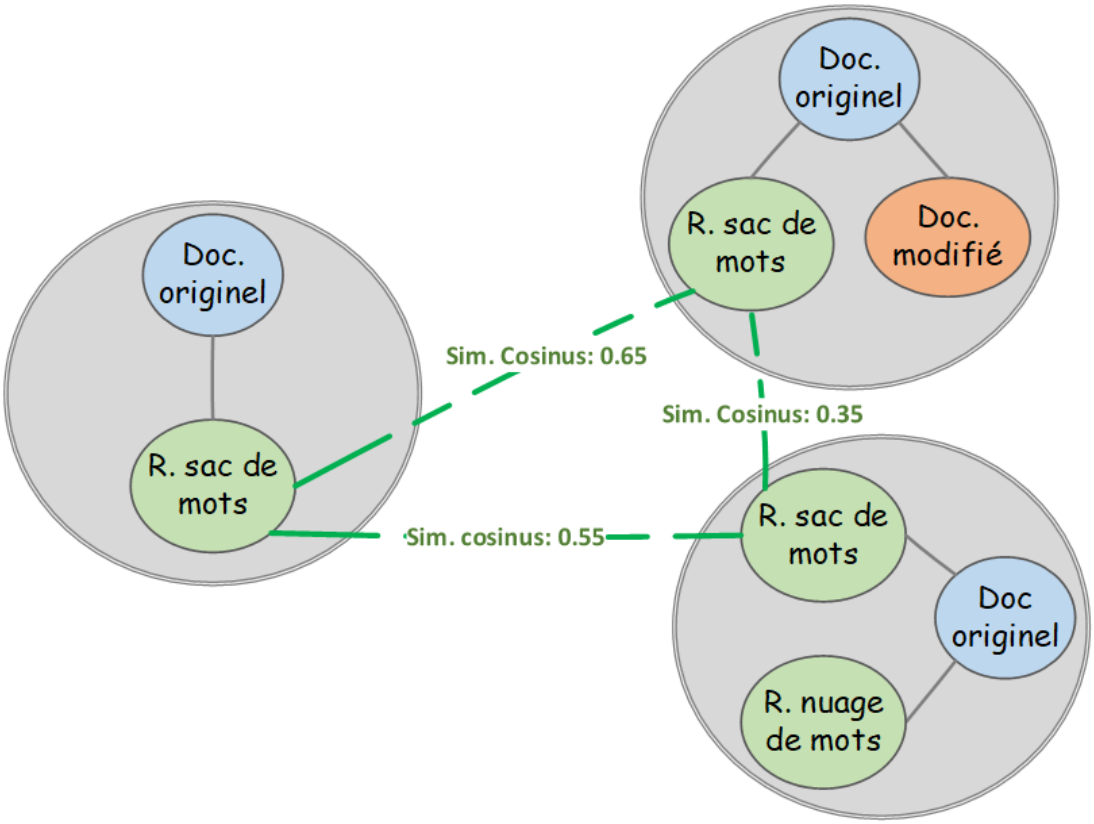


FIGURE 5.6 – Représentation logique des relations de similarité dans MEDAL

5.3.3 Modèle physique de MEDAL

Le niveau logique de MEDAL propose une organisation principalement visuelle. Dans cette section, nous prenons en compte les contraintes d'implémentation physique pour proposer une modélisation physique des données et des métadonnées basée sur le SGBD Neo4J. Ce choix s'explique par le fait que Neo4J passe à l'échelle, contrairement à d'autres technologies orientées sur l'expression des relations comme le *framework* RDF. Notons que cette approche d'implémentation est une alternative de modélisation physique parmi d'autres. Le modèle logique de MEDAL peut en effet être implémenté sur d'autres SGBD orientés graphes.

Représentation physique des métadonnées intra-entité

Dans Neo4J, nous représentons chaque hyper-nœud en même temps que l'entité de données originelle qui lui est associée par un simple nœud sous le label RAW. Ce nœud contient donc à la fois les métadonnées propres à l'entité de données originelle (chemin d'accès, horodatage), mais aussi celles qui sont communes à toute la généalogie (source, titre, etc.). Les nœuds étiquetés REFINED et VERSION correspondent aux représentations et aux versions, respectivement. Il est à noter que Neo4J ne permet pas le stockage de valeurs complexes à l'intérieur des nœuds (listes de clés-valeurs, textes, images, etc.). Par conséquent, les nœuds ainsi définis ne contiennent que des propriétés dont les valeurs sont atomiques. Les valeurs plus complexes sont stockées dans des systèmes de stockage indépendants (système de fichier, SGBD) et référencées par des pointeurs dans Neo4J.

En guise d'illustration, la Figure 5.7 traduit le modèle logique de métadonnées intra-entité présentée dans la Figure 5.3 en modèle physique sous Neo4J.

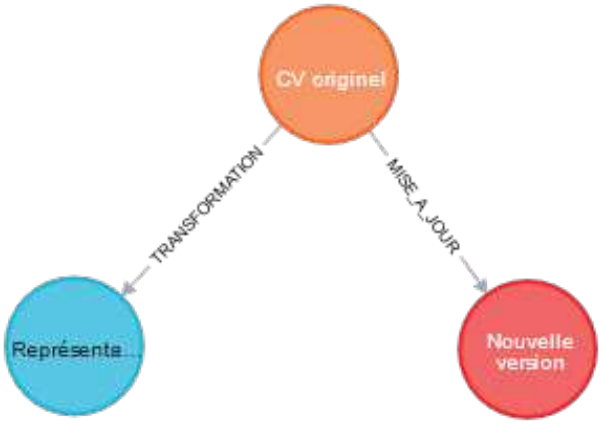


FIGURE 5.7 – Représentation physique des métadonnées intra-entité dans MEDAL

Représentation physique des métadonnées inter-entités

La notion d'hyper-arête n'existe pas dans Neo4J. En guise d'alternative, nous utilisons des nœuds qui servent de *hubs* entre les nœuds liés. Dans le cas d'une liaison de parenté, les nœuds représentant des hyper-arêtes portent le label MERGE. Dans ce cas, le sens des arêtes d'association permet de distinguer les entités de données parentes de l'entité de données enfant (Figure 5.8). De façon analogue, les groupes représentés dans le modèle

logique par des nœuds dentés donnent lieu dans le modèle physique à des nœuds de label GROUP (en vert dans la Figure 5.9). Enfin, les relations de similarité sont représentées par des arêtes dont le label précise le type de similarité utilisé (Figure 5.10). Dans Neo4J, les arêtes sont forcément orientées. La prise en compte ou non de l'orientation se fait au moment de l'interrogation.

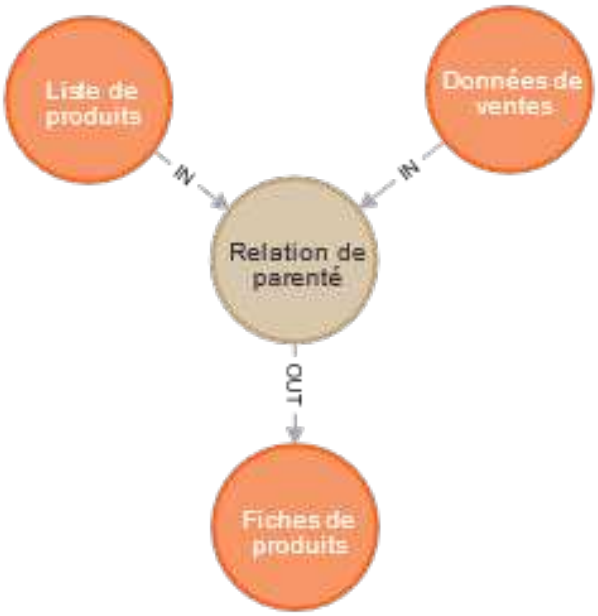


FIGURE 5.8 – Représentation physique des liaisons de parenté

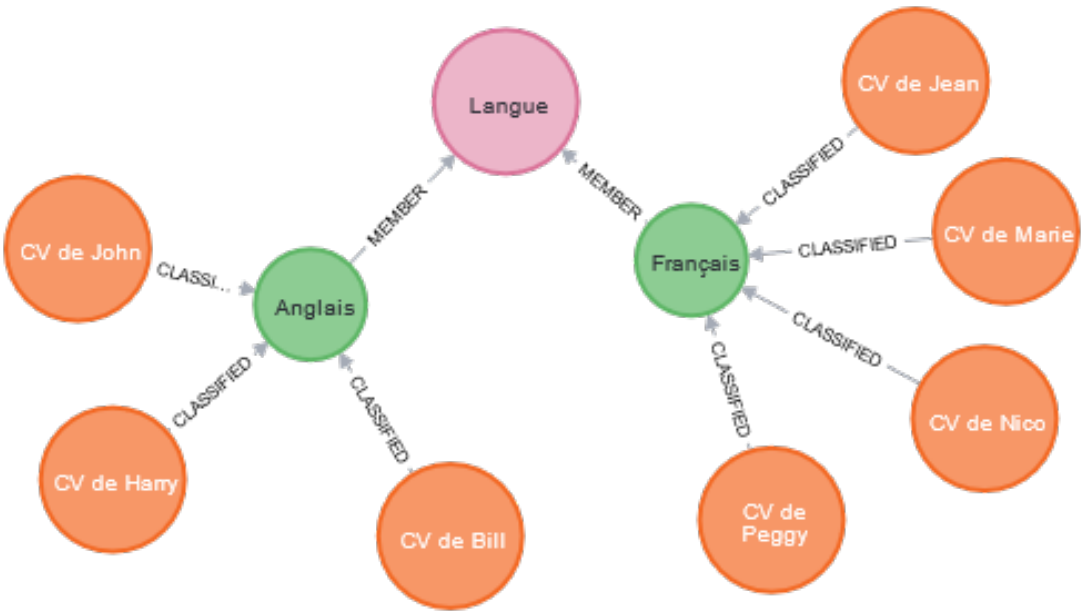


FIGURE 5.9 – Représentation physique des groupements dans MEDAL

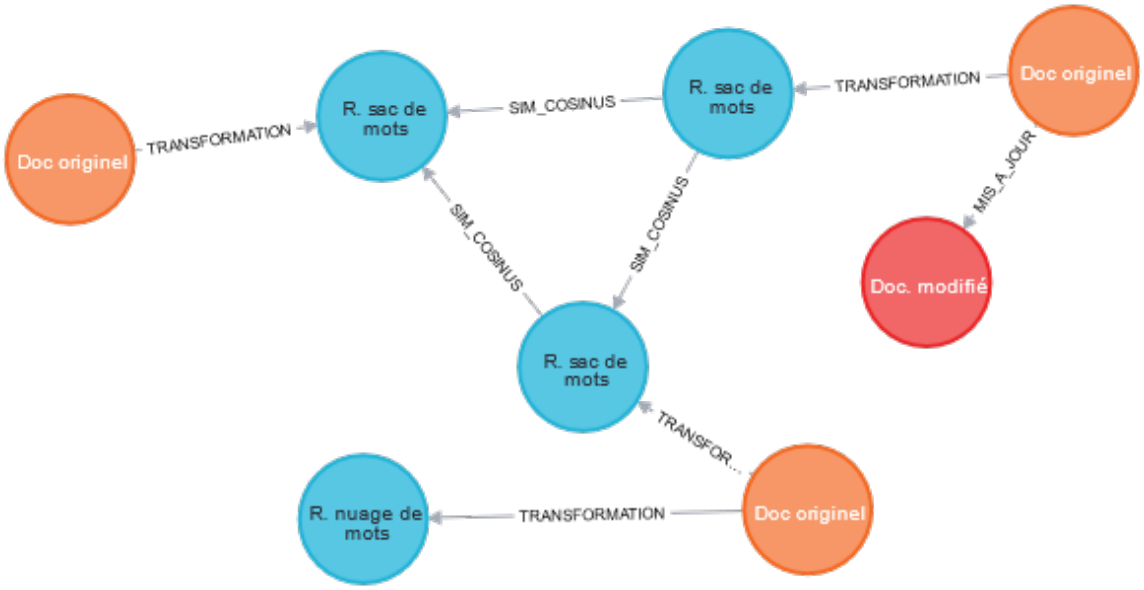


FIGURE 5.10 – Représentation physique des relations de similarité dans MEDAL

5.4 Exemple illustratif

Pour illustrer l'utilisation de MEDAL, nous proposons dans cette section un exemple de cas d'usage, ainsi que les modélisations logiques et physiques correspondantes.

5.4.1 Présentation du cas d'usage

Nous proposons un cas d'usage dérivé du concept de lac de données personnelles (Section 3.3.1). Plus concrètement, nous proposons de répondre au besoin d'organisation et de tri des photos numériques d'une personne à travers un lac de données photographiques. Un tel lac de données permet en effet de traiter plusieurs problématiques liées au stockage et à l'exploitation de données photographiques. Nous en présentons trois.

1. **Recherche de données.** L'utilisateur doit pouvoir retrouver rapidement une photo qui l'intéresse. Pour y parvenir, une solution est de créer des groupements de photos basés sur l'année, la saison, le format de la photo, le type d'appareil utilisé, etc., de sorte à ce que l'utilisateur puisse les filtrer aisément. On pourrait même aller plus loin en créant une catégorie basée sur le type d'objet photographié (personnes, animaux, paysages, repas, monuments, etc.), qui peut à son tour être subdivisée selon la présence ou non de certains objets précis (personnes, animaux, etc.).
2. **Détection de doublons.** Il est fréquent qu'un même objet soit photographié plusieurs fois, en vue d'un tri ultérieur (conservation de la meilleure photo et suppression des doublons). Ce tri peut être facilité par la détection automatisée de photos fortement similaires (relations de similarité). Cela évite à l'utilisateur de rechercher manuellement les potentiels doublons, au risque d'en omettre.
3. **Traçage des représentations.** Pour certains besoins spécifiques, les images peuvent être retraitées par l'utilisateur, à travers par exemple l'application de filtres (représentation en nuances de gris, effet vintage) ou encore le rognage, etc. Toutes ces

retouches donnent lieu à de nouvelles représentations des photos originelles. Il peut arriver aussi que des photos différentes soient fusionnées. Dans ce cas, il est utile de conserver l'information sur le contexte de cette opération. Cela se traduit par une relation de parenté entre les photos fusionnées et la photo résultante.

Dans cet exemple, nous faisons abstraction du versionnement, car cette problématique n'est pas pertinente dans ce cas d'usage. Nous occultons également l'utilisation des métadonnées globales,Dans cet exemple, nous faisons abstraction du versionnement, car cette problématique n'est pas pertinente dans ce cas d'usage. Nous occultons également l'utilisation des métadonnées globales, qui n'induisent pas un réel besoin de modélisation.

5.4.2 Modélisation logique et physique

La Figure 5.11 représente une modélisation logique correspondant à notre lac de données photographiques. Nous y représentons quatre photos à travers quatre nœuds, contenus dans des hyper-nœuds. Nous représentons également un exemple de regroupement suivant le type d'appareil utilisé, ainsi qu'un ensemble de mesures de similarité traduisant les relations de similarité entre les images. Par souci de simplicité, nous omettons le cas de la relation de parenté.

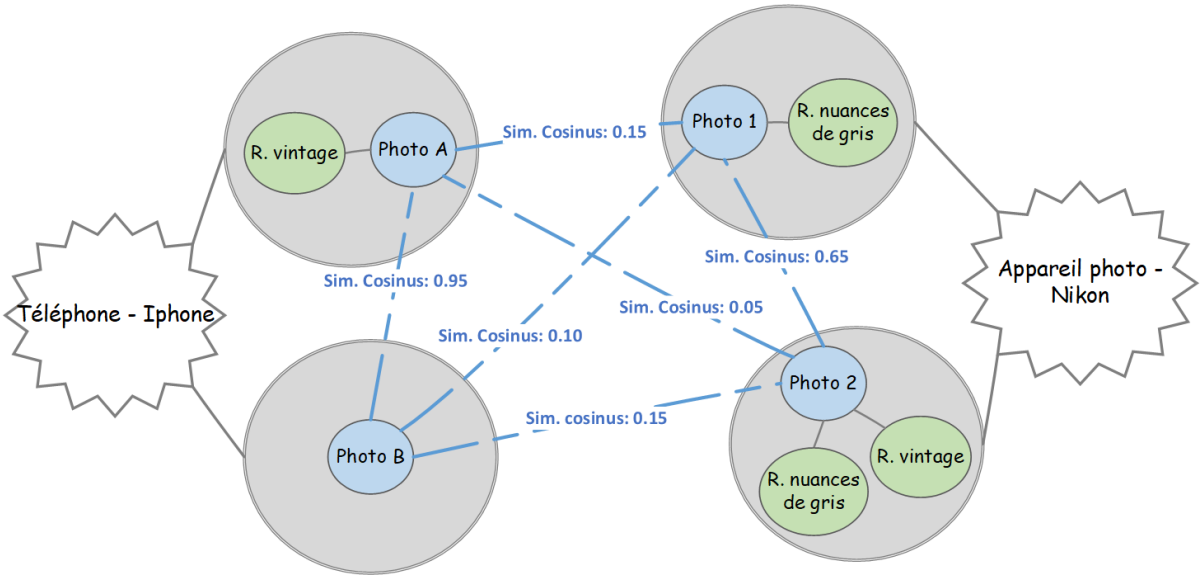


FIGURE 5.11 – Modèle logique du lac de données photographiques

La Figure 5.12 traduit cette représentation logique en implémentation physique sous Neo4J. Les hyper-nœuds, nœuds de représentations et hyper-arêtes du modèle physique y sont traduits en nœuds auxquels sont associés les labels RAW (en orange), REFINED (en bleu) et GROUP (en rouge), respectivement.

5.5 Discussion

Pour évaluer le modèle MEDAL, nous le comparons aux modèles et systèmes de métadonnées existants dans la littérature. Pour ce faire, nous identifions un ensemble de fonctionnalités usuelles des métadonnées dans les lacs de données que nous utilisons comme

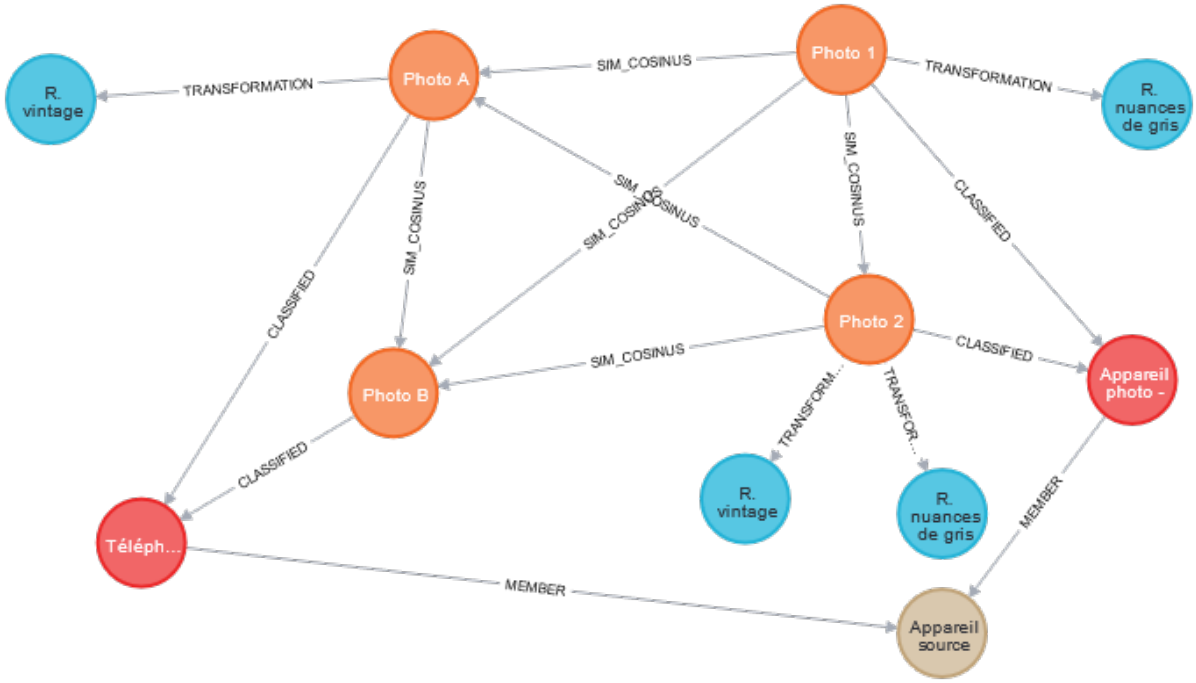


FIGURE 5.12 – Modèle physique du lac de données photographiques

critères de comparaison. Nous considérons ainsi que plus un modèle ou un système supporte de fonctionnalités, plus il peut être considéré comme complet.

5.5.1 Fonctionnalités des systèmes de métadonnées

Nous identifions six fonctionnalités usuelles que doit idéalement supporter le système de métadonnées d'un lac de données.

1. **Enrichissement sémantique (ES)**. Également connue sous l'appellation « annotation sémantique » (HAI et al., 2016) ou « profilage sémantique » (ANSARI et al., 2018), cette notion exprime le fait de pouvoir générer une description du contexte des données, par exemple avec des *tags*, pour leur donner plus de sens et les rendre compréhensibles (TERRIZZANO et al., 2015). L'annotation sémantique se fait généralement à l'aide de bases de connaissances telles que les ontologies. Elle joue un rôle clé dans l'exploitation des données, en résumant le contenu des entités de données du lac, les rendant ainsi plus compréhensibles par les utilisateurs. L'enrichissement sémantique peut également servir de base à l'identification de liens entre les données.
2. **Indexation des données (ID)**. L'indexation consiste à mettre en place une structure de données permettant de retrouver les entités de données sur la base de caractéristiques spécifiques (mots clés ou motifs). Cela nécessite la construction et la prise en charge d'index ou d'index inversés. L'indexation permet d'optimiser l'interrogation des données du lac par filtrage par mots clés. Elle est particulièrement utile pour la gestion des données textuelles, mais peut également être utilisée dans un contexte de données semi-structurées ou même structurées (SINGH et al., 2016).
3. **Génération et conservation de liaisons (GL)**. Cela consiste à détecter de nouvelles connexions ou à intégrer des connexions pré-existantes entre les entités de

données d'un lac. L'intégration de liaisons entre les données permet d'élargir la panoplie d'analyses possibles à partir du lac à travers, par exemple, la recommandation de données liées à celles qui intéressent l'utilisateur (Maccioni & Torlone, 2018). Ces liens peuvent également être utilisés pour identifier des *clusters* d'entités de données, c'est-à-dire des groupes d'entités de données fortement similaires entre elles et significativement différentes des autres (Farrugia et al., 2016).

4. **Polymorphisme des données (PD)**. Nous définissons le polymorphisme des données comme le stockage simultané de plusieurs représentations des mêmes entités de données. Chaque représentation correspond à l'entité de données initiale, modifiée ou reformatée pour un besoin spécifique. Un document textuel peut par exemple être représenté sous la forme d'un sac de mots (vecteur) ou comme un nuage de mots (image). Il est essentiel 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). Il nous paraît donc important de disposer de modèles et de systèmes de métadonnées permettant de stocker simultanément plusieurs représentations raffinées des entités de données ; ceci notamment dans le but d'éviter la répétition des prétraitements et donc d'accélérer les analyses.
5. **Versionnement des données (VD)**. Cette notion fait référence à la capacité du système de métadonnées à prendre en charge les modifications des entités de données tout en conservant les états précédents. En d'autres termes, le versionnement consiste à conserver plusieurs états d'une même entité de données au fil du temps. Cette fonctionnalité est essentielle dans les lacs de données, car elle garantit la reproductibilité des analyses et permet de corriger d'éventuelles erreurs ou incohérences. Le versionnement permet également de supporter une évolution ramifiée des données, notamment dans leur schéma (Hellerstein et al., 2017).
6. **Suivi d'utilisation (SU)**. Le suivi d'utilisation consiste à enregistrer les interactions entre les utilisateurs et le lac de données. Ces interactions sont généralement des opérations de création, de mise à jour et d'accès aux données. L'intégration de ces informations dans le système de métadonnées permet de comprendre et d'expliquer des incohérences éventuelles dans les données (Beheshti et al., 2017). Elle peut également être utilisée pour protéger les données sensibles en détectant les intrusions (Suriarachchi & Plale, 2016).

   Le suivi d'utilisation et le versionnement des données sont étroitement liés, car les interactions conduisent dans certains cas à la création de nouvelles versions ou représentations des données. Cependant, ces deux notions renvoient à des fonctionnalités différentes, d'autant plus qu'elles ne sont pas systématiquement proposées ensemble (Beheshti et al., 2017 ; Diamantini et al., 2018 ; Suriarachchi & Plale, 2016).

5.5.2 MEDAL face aux autres modèles et systèmes de métadonnées

Nous comparons dans cette section les fonctionnalités supportées par MEDAL avec celles de modèles et systèmes de métadonnées de lacs de données de la littérature. Les modèles de métadonnées font référence à des systèmes conceptuels d'organisation des métadonnées.

Ils ont l'avantage d'être plus détaillés et plus facilement reproductibles que les systèmes de métadonnées, qui eux se situent à un niveau plus opérationnel. Nous incluons également dans cette comparaison des systèmes et modèles non explicitement associés au concept de lac de données par leurs auteurs, mais qui peuvent être utilisés dans un contexte de lac de données. C'est le cas du modèle de métadonnées Ground (Hellerstein et al., 2017).

Le tableau 5.1 montre que MEDAL surpasse tous les modèles et systèmes pré-existants. En effet, MEDAL prend en charge l'ensemble des six fonctionnalités identifiées dans la Section 5.5.1.

Table 5.1 – Fonctionnalités des systèmes et modèles de métadonnées

Modèles/Système → **Critères ↓**		ES	ID	GL	PD	VD	SU	-
SPAR (Fauduet & Peyrard, 2010)	♦♯	✓	✓	✓			✓	(4/6)
Alrehamy et Walker (2015)	♦	✓		✓				(2/6)
Terrizzano et al. (2015)	♦	✓	✓			✓	✓	(4/6)
Hai et al. (2016)	♦	✓	✓					(2/6)
GEMMS (Quix et al., 2016)	◊	✓						(1/6)
CLAMS (Farid et al., 2016)	♦	✓						(1/6)
Suriarachchi et Plale (2016)	◊				✓		✓	(2/6)
Singh et al. (2016)	♦	✓	✓	✓	✓			(4/6)
Farrugia et al. (2016)	♦			✓				(1/6)
GOODS (A. Halevy et al., 2016)	♦	✓	✓	✓		✓	✓	(5/6)
CoreDB (Beheshti et al., 2017)	♦		✓				✓	(2/6)
Ground (Hellerstein et al., 2017)	◊♯	✓	✓			✓	✓	(4/6)
KAYAK (Maccioni & Torlone, 2018)	♦	✓	✓	✓				(3/6)
CoreKG (Beheshti et al., 2018)	♦	✓	✓	✓	✓		✓	(5/6)
Diamantini et al. (2018)	◊	✓		✓	✓			(3/6)
MEDAL (Sawadogo et al., 2019b)	◊	✓	✓	✓	✓	✓	✓	(6/6)

♦ : Système de métadonnées ◊ : Modèle de métadonnées
♯ : Modèles et systèmes assimilés

1. **Enrichissement sémantique**. Cette fonctionnalité est supportée dans MEDAL à travers les métadonnées et les ressources sémantiques.
2. **Indexation des données**. MEDAL prend en charge cette fonctionnalité grâce à l'utilisation d'index inversés.
3. **Génération et conservation de liaisons**. MEDAL permet de connecter les données à travers trois types de relations que sont les catégories, les relations de similarité et les liaisons de parenté.

4. **Polymorphisme des données**. La notion de représentation permet de conserver à la fois des données brutes et raffinées dans MEDAL, et donc de supporter le polymorphisme des données.
5. **Versionnement des données**. De façon analogue au polymorphisme des données, le versionnement est assuré grâce aux métadonnées version et à la conservation des opérations de transformation.
6. **Suivi d'utilisation**. Cette fonctionnalité est supportée grâce aux journaux d'évènements et à la conservation des opérations de transformation et de mise à jour.

Outre MEDAL, les systèmes ou modèles les plus complets sont ceux inhérents aux lacs de données GOODS et CoreKG, avec 5 fonctionnalités sur 6 prises en charge. Ils se distinguent des autres à travers le support du polymorphisme et du versionnement des données. Cependant, CoreKG et (surtout) GOODS restent des boîtes noires avec peu de détails sur l'organisation conceptuelle. On peut donc leur préférer Ground, qui est beaucoup plus détaillé et presque aussi complet (4 fonctionnalités sur 6).

En termes de fonctionnalités, on note une quasi-unanimité de la littérature sur la pertinence de l'enrichissement sémantique (12 systèmes sur 15 proposent cette fonctionnalité) et, dans une moindre mesure, de l'indexation des données (9 sur 15) et de la génération de liaisons (8 sur 15). D'autres fonctionnalités sont en revanche beaucoup moins partagées, notamment le polymorphisme des données (4 sur 15) et le versionnement des données (3 sur 15). De notre point de vue, cette rareté ne traduit pas pour autant un manque de pertinence, mais plutôt une complexité d'implémentation. En effet, ces fonctionnalités se retrouvent principalement dans les systèmes les plus complets (GOODS, CoreKG et Ground) et peuvent donc être considérées comme des fonctionnalités avancées.

5.5.3 Avantages et inconvénients de MEDAL

En plus d'être plus complet que les modèles et systèmes de métadonnées pré-existants, MEDAL prend en charge indifféremment tous les types de données (structurées, semi-structurées et non structurées). Par ailleurs, à l'instar d'autres modèles de métadonnées, MEDAL se base sur une approche par graphes permettant de modéliser et d'organiser de façon intuitive et flexible les relations entre les données, mais aussi la dynamique des données (modifications, transformation). Le modèle que nous proposons a aussi l'avantage d'être directif, c'est-à-dire qu'il spécifie de façon précise chaque type de métadonnées à inclure dans le lac de données, ce qui facilite la tâche de construction du lac de données.

Cela nécessite toutefois une multiplicité de concepts qui peuvent compliquer la compréhension du modèle. De plus, MEDAL nécessite pour son implémentation de définir un unique niveau de granularité pour chaque type de données. Par exemple, il ne permet pas d'organiser à la fois les relations entre des tables et des relations entre des n-uplets. Enfin, notre modèle étant basé sur une approche par graphes, il est peu compatible avec les approches traditionnelles de stockage des données basées sur le modèle relationnel. Par conséquent, du fait de la nécessité de systèmes de stockage spécifiques, MEDAL est moins naturel à implémenter que certaines approches comme la modélisation en *data vault* (Section 2.3.2).

5.6 Conclusion

Dans ce chapitre, nous avons détaillé MEDAL, un premier modèle de métadonnées visant la complétude pour la modélisation et l'organisation des métadonnées dans les lacs de données. MEDAL est basé sur une typologie structurelle des métadonnées qui distingue trois catégories de métadonnées, à savoir les métadonnées intra-entité de données, inter-entités et globales.

Au niveau logique, MEDAL propose une organisation basée sur la théorie des graphes à travers quatre concepts principaux. Des nœuds et hyper-nœuds représentent les entités de données et leurs métadonnées intra-entité (descriptions et évolutions), tandis que des arêtes et hyper-arêtes traduisent les métadonnées inter-entités, c'est-à-dire, les relations entre les données. Enfin, les métadonnées globales sont situées au niveau opérationnel uniquement.

Grâce à tous ces éléments, MEDAL prend en charge l'ensemble des six fonctionnalités essentielles que nous avons identifiées pour la gestion des métadonnées dans les lacs de données, ce qui le rend effectivement plus complet que tous les modèles et systèmes de métadonnées pré-existants. MEDAL a aussi l'avantage d'être compatible à la fois avec les données structurées, semi-structurées et non structurées. Cependant, il doit encore être transformé en preuve de concept, ce qui est l'objet du Chapitre 7.1.

Chapitre 6

Modèle de métadonnées goldMEDAL

Sommaire

6.1 Introduction

Dans le chapitre précédent, nous avons proposé MEDAL, un modèle pour l'organisation des métadonnées dans les lacs de données. MEDAL se voulait un modèle à la fois théorique et détaillé, afin de guider les concepteurs de lacs de données dans l'identification et dans l'organisation des métadonnées. Cependant, en catégorisant les entités de données en données brutes (originelles), versions et représentations, nous avons en quelque sorte figé notre modèle de métadonnées, alors que lors de la conception de divers lacs de données, nous avons constaté que d'autres types d'entités de données étaient possibles, par exemple des représentations temporelles.

Pour remédier à cette limite, à l'époque présente dans tous les modèles de métadonnées, nous proposons dans ce chapitre goldMEDAL, une évolution importante de MEDAL sur deux plans complémentaires. En premier lieu, nous avons décidé de placer les concepts de goldMEDAL à un niveau d'abstraction supérieur que ceux de MEDAL. Deuxièmement, nous avons conçu goldMEDAL pour tendre vers un modèle de métadonnées générique, capable de prendre en charge un maximum de cas d'usage, mais tout en restant suffisamment simple pour éviter le syndrome de « l'usine à gaz ».

goldMEDAL est défini à travers les trois niveaux classiques de modélisation : conceptuel, logique et physique. Nous avons choisi une approche formelle pour le modèle conceptuel à des fins de précision, mais nous proposons également un modèle semi-formel (UML) pour une meilleure compréhension, notamment par des non-spécialistes (utilisateurs métiers).

Enfin, pour évaluer goldMEDAL, nous le comparons aux modèles de métadonnées de la littérature sous deux aspects. D'une part, nous illustrons le haut niveau d'abstraction de goldMEDAL en comparant ses concepts à ceux des modèles précédents. D'autre part, nous démontrons son caractère générique en nous basant sur les fonctionnalités prises en charge.

Le modèle goldMEDAL est le résultat d'une collaboration avec Étienne Scholly et Pengfei Liu, doctorant et postdoctorant au laboratoire ERIC, respectivement. Sa description a été publiée sous la référence Scholly, E., Sawadogo, P. N., Liu, P., Espinosa-Oviedo, J.-A., Favre, C., Loudcher, S., Darmont, J. & Noûs, C. (2021b). 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*, 31-40. Une version étendue de cet article a également été soumise pour publication dans la revue internationale *Information Systems*.

6.2 Modèle de métadonnées goldMEDAL

Dans cette section, nous détaillons d'abord les éléments utilisés par goldMEDAL au niveau conceptuel. Nous présentons ensuite leurs traductions au niveau logique à travers la théorie des graphes. Enfin, nous illustrons une implémentation physique de ces concepts à l'aide du SGBD orienté graphes Neo4J.

6.2.1 Modèle conceptuel de goldMEDAL

Au niveau conceptuel, goldMEDAL organise les données à travers quatre éléments de base à savoir les entités de données, les groupements, les liens et les processus. Chacun de ces éléments porte des métadonnées, sous la forme de propriétés (attributs atomiques) ou de listes de propriétés de type clés-valeurs.

Entités de données

Dans goldMEDAL, une entité de données représente à la fois des données brutes, mais également des données modifiées ou transformées (versions et représentations dans MEDAL). Au sens de goldMEDAL, le concept d'entité de données est flexible en termes de granularité. Ainsi, il peut à la fois représenter une base de données, une table relationnelle ou encore un n-uplet. C'est ce qui permet à goldMEDAL de supporter simultanément des données de niveaux de granularité hétérogènes.

Définition 6.2.1. L'ensemble des entités de données est noté $\mathcal{E} = \{e_i\}_{i \in \mathbb{N}^*}$.

Groupements

Un groupement consiste en un ensemble de groupes qui servent à catégoriser les entités de données en collections.

Définition 6.2.2. L'ensemble des groupements est noté $\mathcal{G} = \{G_j\}_{j \in \mathbb{N}^*}$, avec $G_j = \{\Gamma_{jk}\}_{k \in \mathbb{N}^*}$, $\Gamma_{jk} \subseteq \mathcal{E}$ étant un groupe.

Les groupements permettent d'émuler diverses composantes des lacs de données au niveau des métadonnées, comme par exemple, pour les plus classiques de la littérature :

- le **groupement par zones** reprend l'idée de LaPlante et Sharma (2016) d'organiser les données dans le lac en zones, selon leur niveau de raffinement. Dans goldMEDAL, ces zones sont traitées comme des groupes ;
- le **groupement par bassins** correspond à une séparation des données du lac selon leurs types. Chaque type de données donne lieu à un groupe dont les éléments peuvent être stockés, traités et analysés de façon uniforme. Ce groupement est en réalité une application de l'idée de B. Inmon (2016) de séparer les données du lac en bassins de données (Section 4.2.1) ;
- le **groupement par granularité** consiste à catégoriser les entités de données en fonction de leur niveau de granularité. Ce groupement peut aussi exprimer une hiérarchie constituée, par exemple, d'un groupe pour le niveau « base de données », d'un autre pour le niveau « table relationnelle » et d'un autre encore pour le niveau « n-uplet » ;
- le **groupement sémantique** consiste à rassembler des entités de données portant plus ou moins les mêmes informations. Plus concrètement, l'ensemble de la généalogie (versions et représentations) d'une entité de données constitue un groupe sémantique.

Outre ces groupements courants, goldMEDAL peut prendre en charge tout autre type de groupements, comme des groupements basés sur la langue ou l'origine des données. De

façon formelle, un groupement par langue pourrait s'écrire $G_1 = \{\Gamma_{11}, \Gamma_{12}\}$, avec Γ_{11} et Γ_{12} représentant les groupes « français » et « anglais », respectivement.

Liens

Les liens permettent d'associer soit des entités de données entre elles, soit d'établir une relation entre des groupes. Les liens entre entités de données servent principalement à exprimer des relations de similarité. Ils peuvent être orientés ou non. Les liens entre groupes expriment quant à eux des des relations hiérarchiques.

Définition 6.2.3. L'ensemble des liens est noté $\mathcal{L} = \{l_m\}_{m \in \mathbb{N}^*}$, avec soit :

— $l_m : \mathcal{E} \to \mathcal{E}$ (lien entre entités de données),

— $l_m : G_j \to G_{j'}$ and $j \neq j'$ (lien entre groupes).

Par exemple, on peut imaginer une relation hiérarchique entre les groupes associés à un groupement par année et ceux qui sont associés à un groupement par trimestre. Une telle relation est exprimée à l'aide de liens associant chaque trimestre à l'année concernée.

Soient $G_1 = \{2019, 2020, 2021...\}$ le groupement des entités de données par année et $G_2 = \{T1\_2019, T2\_2019, T3\_2019...\}$ le groupement par trimestre. Le lien l_1 permet de représenter formellement la relation hiérarchique entre les groupes de G_1 et G_2. On obtient donc $T1\_2019 \underset{l_1}{\to} 2019, T1\_2019 \underset{l_1}{\to} 2019, T3\_2019 \underset{l_1}{\to} 2019$, etc. Inversement, on peut également noter $2019 \underset{l_1^{-1}}{\longrightarrow} \{T1\_2019, T2\_2019, T3\_2019, T4\_2019\}$.

Processus

Un processus représente une opération de transformation ou de mise à jour ayant entraîné la création d'une nouvelle entité de données. Il permet de tracer les relations généalogiques entre des entités de données. Chaque instance de processus connecte ainsi une ou plusieurs entités de données « parents » à une entité de données « enfant ».

Définition 6.2.4. L'ensemble des processus est noté $\mathcal{P} = \{P_n\}_{n \in \mathbb{N}^*}$, avec $P_n = \{I_n, O_n\}$, $I_n \subseteq \mathcal{E}$ l'ensemble des entités de données en entrée de P_n et O_n l'ensemble des entités de données en sortie, qui est intégré à $\mathcal{E}$ ($\mathcal{E} \leftarrow \mathcal{E} \cup O_n$).

Contrairement aux liens, les processus traduisent des informations complexes. Ils représentent le contexte d'exécution d'une opération de transformation ou de modification (utilisateur, script utilisé, etc.).

La Figure 6.1 récapitule les concepts de goldMEDAL et leurs interactions sous la forme d'un modèle de classes UML. Les entités de données y sont présentées par une classe centrale à laquelle sont associées des classes traduisant les processus, liens et groupements.

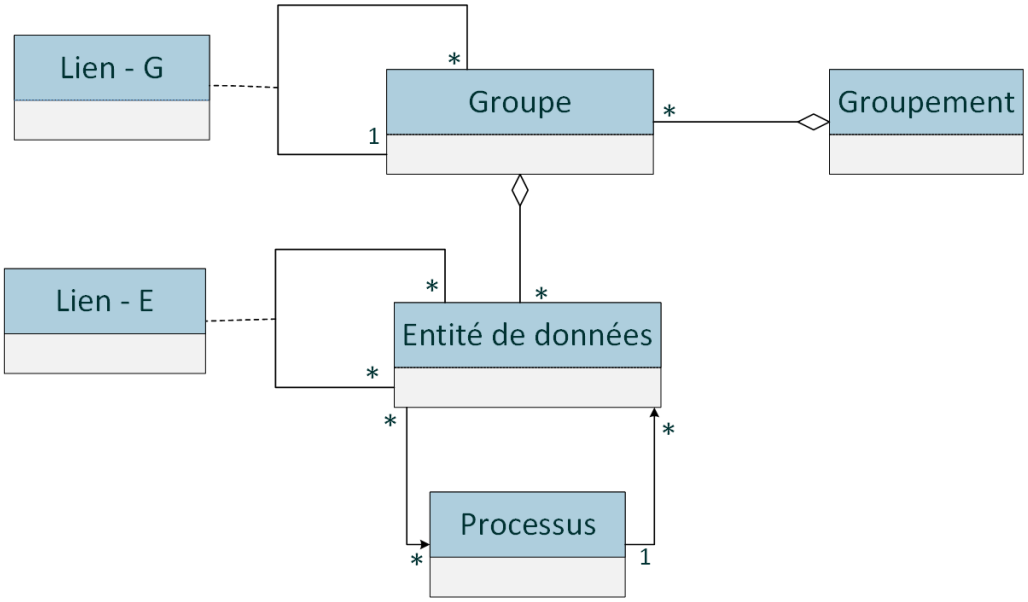


FIGURE 6.1 – Modèle conceptuel de goldMEDAL

6.2.2 Modèle logique de goldMEDAL

Comme celui de MEDAL, le niveau logique de goldMEDAL est basé sur la théorie des graphes. Il consiste donc à traduire les concepts de la Section 6.2.1 en concepts de graphes. goldMEDAL exploite ainsi les notions de nœud, d'arête et d'hyper-arête pour représenter les concepts définis dans le modèle conceptuel (entités de données, groupements, liens et processus).

Représentation logique des entités de données

Dans le modèle logique de goldMEDAL, chaque entité de données se traduit simplement par un nœud. Ce nœud symbolise à la fois l'entité de données elle-même, ainsi que les métadonnées qui lui sont directement associées. Ces propriétés sont par exemple le titre, la date de création, la date de dernière modification, etc. Sur un plan visuel, chaque nœud est représenté par un cercle.

Représentation logique des groupements

Comme dans MEDAL, les groupements se traduisent dans le modèle logique de goldMEDAL par des ensembles d'hyper-arêtes, c'est-à-dire des arêtes qui associent plusieurs nœuds. Chaque hyper-arête représente alors un groupe. Les hyper-arêtes peuvent être représentées visuellement soit par une surface qui englobe l'ensemble des nœuds concernés, soit par un cercle denté auquel sont connectés tous les nœuds associés.

Les Figures 6.2 et 6.3 illustrent ces deux types de représentations à travers l'exemple d'une catégorisation de documents par service (comptabilité et ressources humaines, respectivement). Dans la suite de ce chapitre, nous aurons une préférence pour la deuxième approche, car elle est facilement adaptable pour la représentation d'hyper-arêtes orientées. Cela nous permet de représenter les processus de façon quasi-analogue aux groupements

et de rester en ligne avec l'approche adoptée dans MEDAL.

Représentation logique des liens

Les liens sont représentés dans le modèle logique par de simples arêtes, orientées ou non. Les arêtes exprimant une équivalence ou une similarité sont généralement non-orientées, parce que l'information portée par l'arête est identique dans les deux sens. Nous représentons ce type d'arêtes par des lignes discontinues. Les arêtes exprimant des relations dirigées sont en revanche toujours orientées. C'est typiquement le cas des relations hiérarchiques entre des groupes.

La Figure 6.4 illustre la représentation visuelle des liens entre groupes et entités de données. Nous y représentons, d'une part, des relations de similarité entre des documents (lignes discontinues et non orientées en bleu) et, d'autre part, des relations hiérarchiques entre des groupes (lignes continues et orientées en orange).

Représentation logique des processus

Les processus se traduisent au niveau logique par des hyper-arêtes orientées. Chacune de ces hyper-arêtes sert à associer l'ensemble des nœuds parents à un unique nœud enfant. Visuellement, nous représentons cette hyper-arête orientée par un cercle denté central auquel les nœuds parents et le nœud enfant sont associés. Une flèche permet de distinguer le nœud enfant des autres.

La Figure 6.5 illustre la représentation logique des processus dans goldMEDAL. Elle présente deux exemples de processus, dont le premier symbolise la fusion de deux documents tabulaires, tandis que le deuxième représente une opération de raffinement appliquée à un document textuel.

6.2.3 Modèle physique de goldMEDAL

Le niveau de modélisation physique consiste à implémenter le modèle logique suivant une technologie particulière. Comme pour MEDAL, nous basons les exemples de modélisation physique de goldMEDAL sur le SGBD orienté graphes Neo4J. Le choix de Neo4J s'explique par le fait qu'il est particulièrement adapté pour implémenter les concepts issus de la théorie des graphes en plus de passer à l'échelle.

Dans cette proposition de représentation physique, les entités de données sont des nœuds de Neo4j étiquetés ENTITY (en orange). Ces nœuds stockent les propriétés associées aux entités de données (titre, date de création, identifiants, etc.). Pour représenter les groupements, chaque groupe est traduit en un nœuds portant le label GROUP (en rouge). Chaque nœud d'entité de données est associé aux nœuds des groupes lui correspondant par une arête étiquetée du nom du groupement. Une entité de données peut par exemple être associée au groupe « 2021 » par une arête étiquetée « Année » et à un autre groupe « 1er Trimestre 2021 » par une autre arête étiquetée « Trimestre » ; avec « 2021 » et « 1er Trimestre 2021 » étant des groupes issus de deux groupements distincts.

Les processus, qui sont modélisés par des hyper-arêtes dans le modèle logique, se traduisent par des nœuds dans Neo4J. Ces nœuds sont en réalité des pointeurs vers des scripts (R,

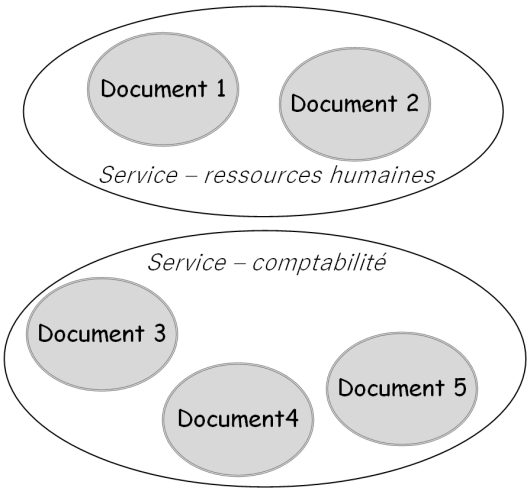


FIGURE 6.2 – Hyper-arêtes en cadres

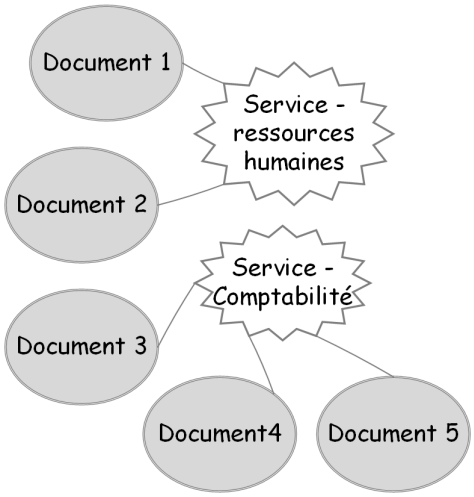


FIGURE 6.3 – Hyper-arêtes en hubs

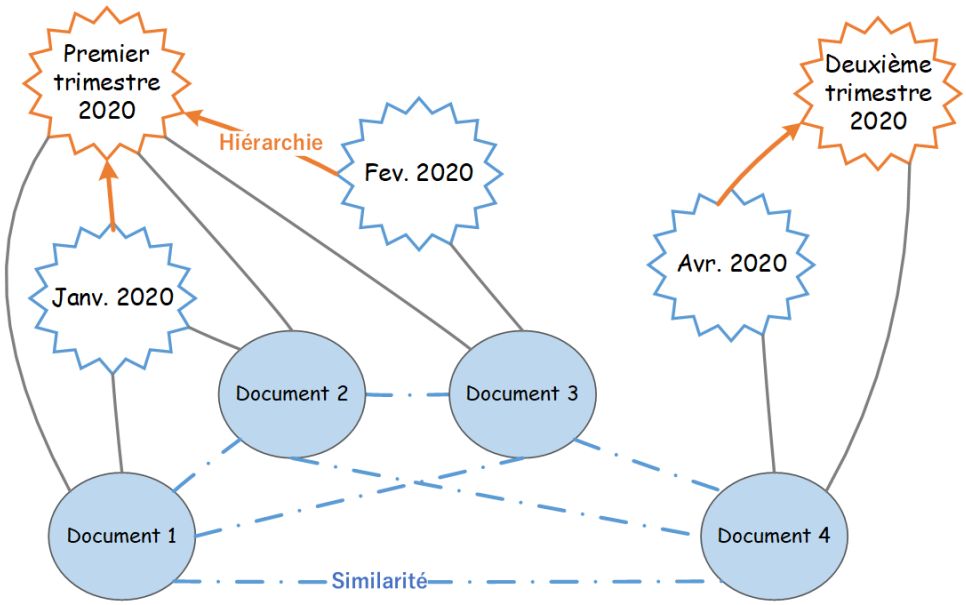


FIGURE 6.4 – Représentation logique de liens

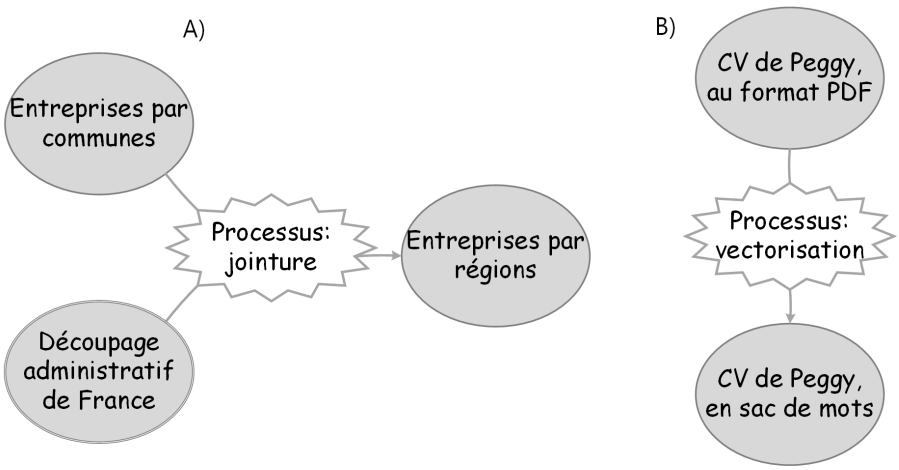


FIGURE 6.5 – Representation logique de processus

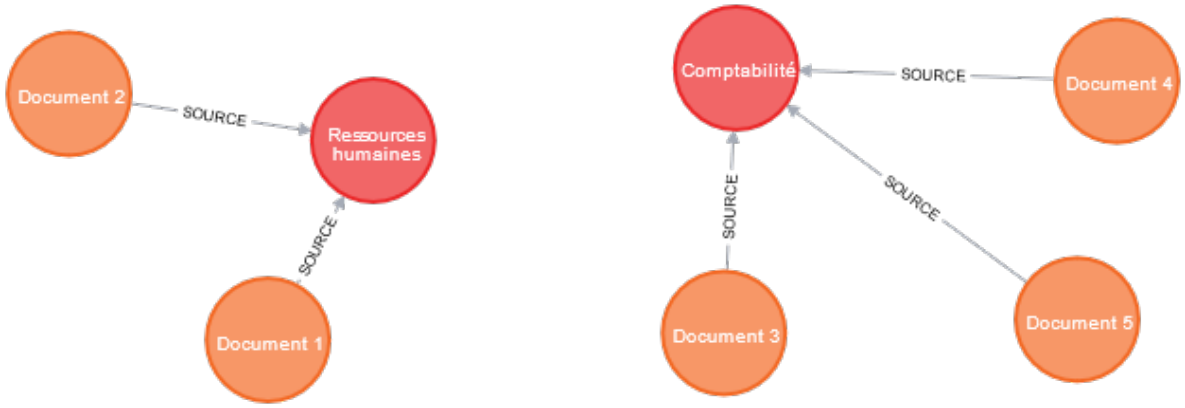


FIGURE 6.6 – Représentation physique d'un groupement

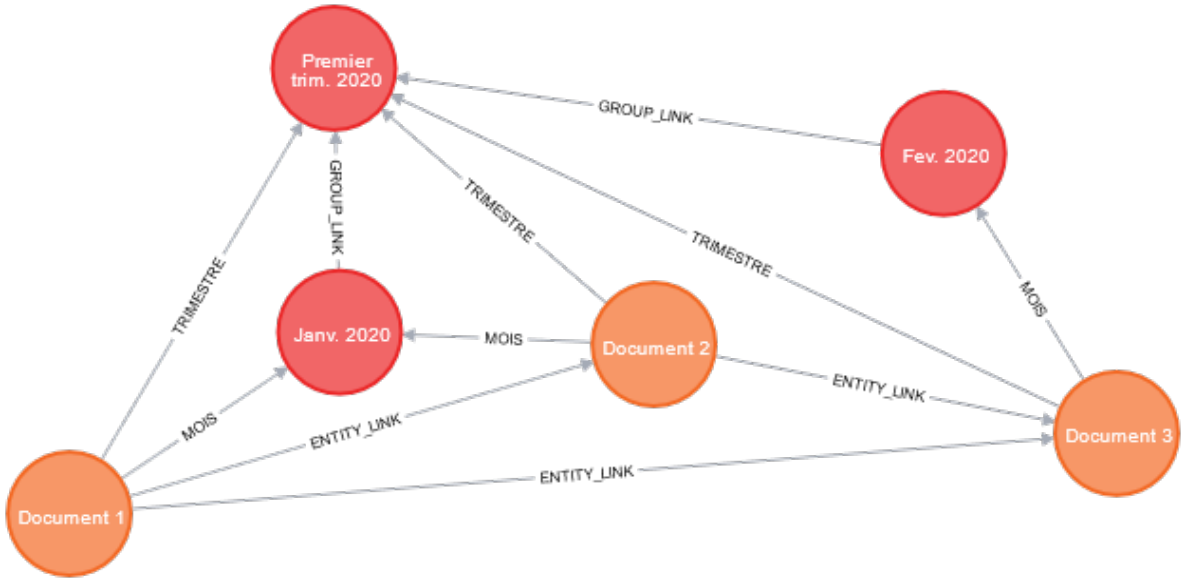


FIGURE 6.7 – Représentation physique de liens

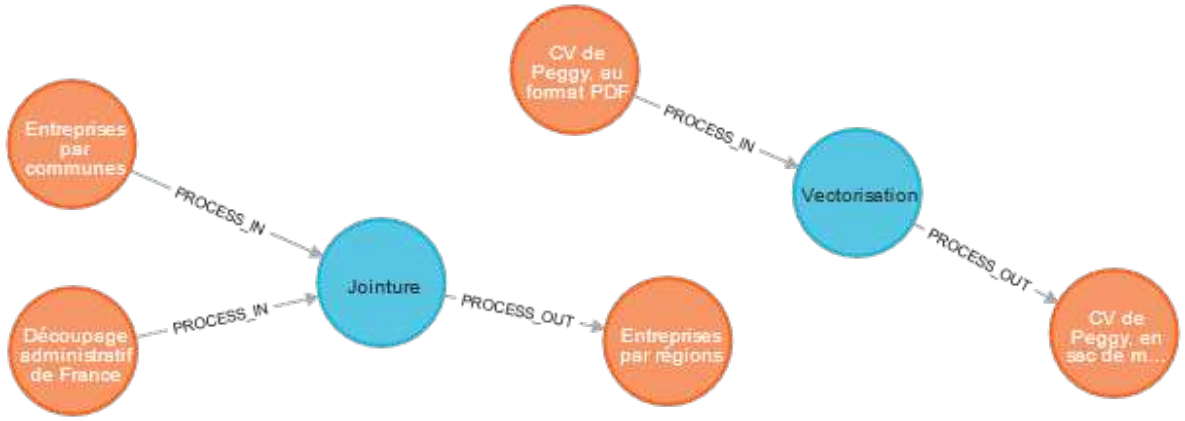


FIGURE 6.8 – Représentation physique de processus

Python, notebook...) utilisés pour réaliser l'opération traduite par le processus. Ils sont étiquetés PROCESS. Des arêtes intitulées PROCESS IN relient les nœuds des entités de données en entrée au nœud du processus. Inversement, une arête intitulée PROCESS OUT relie le processus aux entités de données en sortie.

Enfin, les liens sont représentés par des arêtes auxquelles on associe les labels ENTITY LINK ou GROUP LINK, selon qu'ils associent des nœuds d'entités de données ou des nœuds de groupes.

Les Figures 6.6, 6.7 et 6.8 représentent des exemples d'implémentations physiques de groupes, de liens et de processus, respectivement. Ce sont des traductions vers le modèle physique des représentations logiques des Figures 6.3, 6.4 et 6.5. Dans le cas spécifique de la Figure 6.7, nous ne représentons qu'une sous-partie du modèle physique correspondant, par souci de lisibilité.

6.3 Application à un lac de données locatives

Pour mieux illustrer la mise en œuvre du modèle goldMEDAL, nous présentons dans cette section son utilisation dans un cas d'usage industriel tiré des travaux de Scholly et al. (2021a), avec qui nous avons collaboré lors de la conception de goldMEDAL.

6.3.1 Présentation du lac de données locatives

Contexte

Avec l'émergence des réseaux sociaux, de l'Internet des objets et surtout des plateformes de données ouvertes, les entreprises ont de plus en plus de données à disposition. Ces données constituent une opportunité pour les entreprises d'améliorer leur rendement. De façon plus spécifique, ces données aident les bailleurs de logements à mieux rentabiliser leur activité grâce à des analyses prédictives, sur la base de méthodes d'intelligence artificielle.

Plus concrètement, l'analyse des données permet aux bailleurs de prédire la présence de dégradations (naturelles ou sous l'action humaine) dans un logement. Cette information est particulièrement utile, car elle sert à déterminer l'opportunité (ou non) de réaliser une visite de sortie pour un logement donné. En jugeant d'avance de la nécessité de réaliser une visite de sortie à un locataire, le bailleur peut ainsi économiser dans certains cas les dépenses liées à cette visite (déplacement d'un employé, honoraires d'une agence). Grâce à ces informations, les visites de sorties et les dépenses relatives ne peuvent donc plus réalisées que dans les cas où il y a de réelles chances de trouver des dégradations. La prédiction de la présence de dégradations dans les logements permet également au bailleur de commander d'avance les travaux à réaliser. Cela procure le double bénéfice de minimiser le coût des travaux et les périodes de vacance des logements et, par conséquent, le manque à gagner en termes de loyer.

Problématique et objectifs

Le projet « État des lieux » (EDL) a été proposé pour mettre en œuvre et systématiser ce type d'analyses. Il exploite un ensemble de données locatives aux formats principa-

lement structurés (feuilles de calcul, fichiers CSV) à partir desquelles les *data scientists* réalisent des traitements. Ces traitements consistent typiquement à préparer les données (nettoyage, sélection, fusion, etc.), puis à leur appliquer des algorithmes d'intelligence artificielle pour obtenir les prédictions recherchées. La mise en œuvre effective du projet fait face à trois problématiques importantes qui peuvent être résolues par la mise en place d'un lac de données.

1. **Diversité des analyses.** Plusieurs types d'analyses sont réalisées dans le cadre du projet EDL. Des analyses simples, de type OLAP, sont ainsi effectuées en même temps que des analyses de *machine learning*, plus avancées. Pour prendre en charge l'ensemble de ces analyses, les données doivent être stockées suivant une approche flexible permettant de les modifier et de les restructurer à souhait. C'est justement ce que propose le concept de lac de données, à travers l'approche de stockage *schema-on-read*.
2. **Multiplicité des formats de données.** Au fil des analyses, des données nouvelles sont créés dans le projet EDL. Cela inclue des *dataframes* aux formats RData ou pkl, des scripts au format R ou Jupyter Notebook, ou encore des images, présentations et rapports. Face à cette diversité, la recherche de données devient une tâche particulièrement complexe. Une solution consiste à organiser les données selon leur format et leur niveau de raffinement, qui revient à appliquer les principes de bassins et de zones inhérents aux lacs de données.
3. **Traçage de la généalogie des données.** Les analyses et traitements étant réalisés par des équipes différentes, un suivi de la généalogie des données est nécessaire pour les rendre compréhensibles et donc utilisables par l'ensemble des équipes. Cela permet plus concrètement de savoir par qui (quelle équipe ou quel utilisateur) et comment (quelles données sources, quels traitements) des données nouvelles sont été générées. Un système de versionnement des données est nécessaire pour cela.

6.3.2 Modèles logique et physique

Pour faire face à ces problématiques et mettre en œuvre le projet EDL, la solution d'un lac de données a été adoptée. Ce lac de données, intitulé HOUDAL (*Public HOUsing Data Lake*), a été implémenté en s'appuyant sur goldMEDAL. Seul le concept de lien n'y est pas utilisé.

La Figure 6.9 représente un exemple de modélisation logique correspondant aux métadonnées de HOUDAL. Nous y présentons trois entités de données, dont l'une est le résultat de la fusion des deux autres. Nous ajoutons dans cette représentation logique un groupement selon le format et un autre suivant le niveau de raffinement des données. La Figure 6.10 traduit cette représentation logique en une représentation physique sous Neo4J.

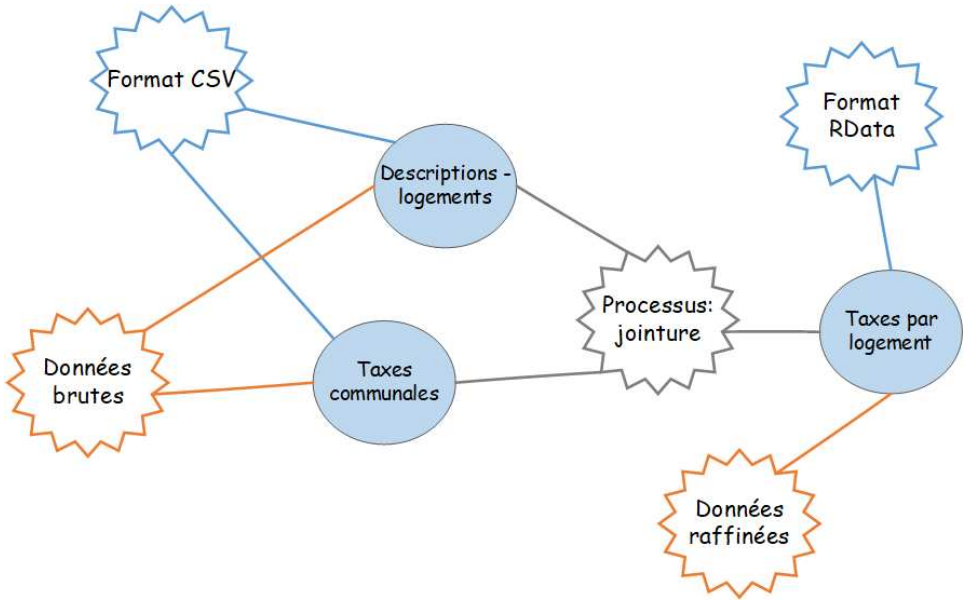


FIGURE 6.9 – Modèle logique des métadonnées de HOUDAL

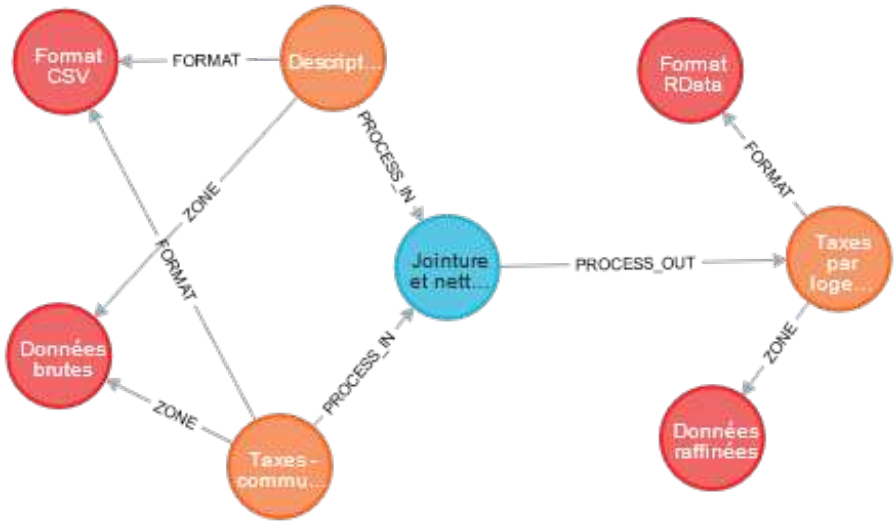


FIGURE 6.10 – Modèle physique des métadonnées de HOUDAL

6.4 Discussion

Dans cette section, nous analysons le modèle goldMEDAL sous deux aspects. Dans un premier temps, nous étudions la généricité de goldMEDAL à travers une analyse des fonctionnalités supportées. Dans un second temps, nous démontrons le haut niveau d'abstraction adopté par goldMEDAL en le comparant à ses prédécesseurs.

6.4.1 Analyse de la généricité de goldMEDAL

Critères d'évaluation de la généricité

Nous définissons la généricité d'un modèle de métadonnées par sa capacité à prendre en charge un maximum de cas d'usage. Un modèle parfaitement générique doit donc supporter tous les cas d'usage possibles d'un lac de données. Étant donné que chaque cas d'usage nécessite des fonctionnalités spécifiques de gestion des métadonnées, nous considérons que ces fonctionnalités peuvent servir de base à l'évaluation de la généricité des modèles de métadonnées. Ainsi, plus un modèle de métadonnées supporte de fonctionnalités, plus il est générique.

À notre connaissance, il n'existe dans la littérature que deux comparaisons des modèles de métadonnées sur la base de fonctionnalités. La première est celle que nous avons proposée dans la section 5.5.1 pour comparer les systèmes et modèles de métadonnées de lacs de données. Pour rappel rappel, nous y avions identifiées six fonctionnalités pertinentes :

1. l'enrichissement sémantique,
2. l'indexation des données,
3. le polymorphisme des données,
4. le versionnement des données,
5. la génération et la conservation de liaisons,
6. le suivi d'utilisation.

La deuxième comparaison sur la base de fonctionnalités est l'œuvre d'Eichler et al. (2020). Dans cette analyse, les auteurs se basent sur trois fonctionnalités pour juger de la généricité des modèles de métadonnées de lacs de données.

1. Le **support des métadonnées de type propriétés** désigne la capacité du modèle de métadonnées à considérer et à intégrer les métadonnées atomiques associées aux entités de données. Ce sont par exemple le titre, les horodatages de création ou de modification, l'URL de stockage, etc. Cette fonctionnalité est supportée dans la quasi-totalité des modèles de métadonnées, à l'exception du modèle de Diamantini et al. (2018), qui est centré sur l'expression des relations entre les données.
2. L'**organisation en zones** consiste pour le modèle de métadonnées à prendre en charge la coexistence de données de niveaux de raffinement différents. Plus concrètement, le modèle doit permettre de distinguer les données brutes des données qui sont plus raffinées. Cette fonctionnalité est l'une des innovations apportées par le modèle de Ravat et Zhao (2019a).

3. Les **niveaux de granularité multiples** permettent au modèle de métadonnées de prendre en charge simultanément plusieurs niveaux de granularité des mêmes entités de données. Par exemple, le modèle doit pouvoir décrire un n-uplet en même temps qu'une table relationnelle et une base de données relationnelle au sein d'une hiérarchie. Cette fonctionnalité est prise en charge par GEMMS (Quix et al., 2016), le premier modèle de métadonnées dédié aux lacs de données.

Considérant que les deux ensembles de caractéristiques sus-cités sont tous pertinents, nous proposons de les combiner pour comparer la généricité des modèles de métadonnées et ainsi évaluer la généricité de goldMEDAL. Au-delà de la fusion des deux ensembles de fonctionnalités nous introduisons trois amendements. D'abord, nous fusionnons la fonctionnalité de polymorphisme des données avec celle relative à l'organisation en zones, car elles font toutes deux référence au même concept. Ensuite, nous divisons la fonctionnalité de génération et de conservation de liaisons en deux fonctionnalités distinctes, à savoir la prise en charge de relations de similarité et le support de catégorisations. Ce choix se justifie par le fait que certains modèles de métadonnées ne prennent en charge qu'une seule de ces deux (sous) fonctionnalités. Enfin, nous omettons la fonctionnalité d'indexation des données car elle est intimement liée à la mise en œuvre opérationnelle du système de métadonnées. Il s'agit donc d'une question technique plutôt qu'un besoin de modélisation. Par conséquent, nous considérons que l'indexation, si elle demeure pertinente pour comparer des systèmes de métadonnées (comme nous l'avons fait dans la section 5.5.1), est inadaptée en ce qui concerne le cas spécifique des modèles de métadonnées.

En résultat, nous obtenons un ensemble de huit fonctionnalités permettant d'évaluer et de comparer la généricité des modèles de métadonnées :

1. enrichissement sémantique,
2. polymorphisme des données ou organisation en zones,
3. versionnement des données,
4. suivi d'utilisation,
5. prise en charge de catégorisations,
6. prise en charge de relations de similarité,
7. support des métadonnées « propriétés »,
8. prise en charge de multiples niveaux de granularité.

Généricité de goldMEDAL

La Table 6.1 présente une comparaison des fonctionnalités prises en charge par goldMEDAL et autres modèles de métadonnées. Elle montre que seul goldMEDAL prend en charge l'ensemble des huit fonctionnalités. Dans ce qui suit, nous détaillons la façon dont chaque fonctionnalité est prise en charge dans goldMEDAL.

1. L'**enrichissement sémantique** est pris en charge par goldMEDAL à travers les liens entre groupes. Ces liens permettent en effet d'exprimer non seulement des relations hiérarchiques, mais aussi des équivalences entre groupes. Ces relations peuvent ainsi servir à désambiguïser et à enrichir les groupements sémantiques.

TABLE 6.1 – Généricité des modèles de métadonnées

Modèles → Fonctionnalités ↓	*GEMMS*	*Ground*	*Diamantini*	*Ravat & Zhao*	*MEDAL*	*HANDLE*	*goldMEDAL*
Enrichissement sémantique	✓	✓	✓	✓	✓	✓	✓
Multiples zones			✓	✓	✓	✓	✓
Versionnement		✓		✓	✓		✓
Suivi d'utilisation		✓		✓	✓	✓	✓
Catégorisations	✓	✓		✓	✓	✓	✓
Liaisons de similarité			✓	✓	✓	✓	✓
Metadonnées propriétés	✓	✓		✓	✓	✓	✓
Multiple granularités	✓		✓			✓	✓
-	(4/8)	(5/8)	(4/8)	(7/8)	(7/8)	(7/8)	(8/8)

2. L'**organisation en zones** est prise en charge dans goldMEDAL à travers le concept de groupements. Chaque entité de données est ainsi associée à un groupe issu d'un groupement en zones.
3. Le **versionnement des données** est pris en charge grâce aux processus, qui permettent de tracer la généalogie des données. La gestion du versionnement se fait à l'aide de processus de type « mise à jour ».
4. Le **suivi d'utilisation** est également pris en charge par goldMEDAL à travers les processus, qui permettent en effet de savoir qui utilise les données du lac et quels traitements sont réalisés.
5. Les **catégorisations** sont prises en charge par le concept de groupement. D'ailleurs, contrairement à certains modèles (HANDLE, modèle de RAVAT et ZHAO (2019a)), goldMEDAL permet d'aller au delà d'une unique instance de catégorisation. Comme avec MEDAL, un nombre illimité de catégorisations est possible.
6. Les **relations de similarité** sont prises en charge à travers le concept de lien. Les liens permettent non seulement de connecter des entités de données, mais aussi de lier des groupes.
7. Les **métadonnées « propriétés »** sont prises en charge à travers les entités de données, qui représentent à la fois les ensembles de données (brutes ou raffinées) et les descriptions qui leur sont associées.
8. Les **granularités multiples** sont prises en charge par le concept de groupement, à l'image de l'organisation en zones. Dans le cas présent, chaque niveau de granularité se traduit par un groupe auquel les entités de données relatives sont associées.

À part goldMEDAL, les modèles les plus génériques sont HANDLE (Eichler et al., 2020), le modèle de Ravat et Zhao (2019a) et MEDAL (Sawadogo et al., 2019b). Cela démontre une tendance croissante à la généricité dans les modèles de métadonnées de lacs de données.

6.4.2 Analyse du niveau d'abstraction de goldMEDAL

Pour illustrer le haut niveau d'abstraction adopté dans goldMEDAL, nous comparons ses concepts à ceux de la littérature. Nous montrons ainsi que goldMEDAL permet de conceptualiser la quasi-totalité des notions utilisées par les modèles de métadonnées existants avec un nombre minimal de concepts. Dans cette comparaison, nous nous focalisons sur HANDLE, le modèle de Ravat et Zhao (2019a) et MEDAL, qui sont les plus génériques et les plus récents.

Concepts de goldMEDAL *vs.* concepts du modèle de Ravat & Zhao

Les concepts de goldMEDAL constituent une généralisation des concepts définis par Ravat et Zhao (2019a). Le concept d'entité de données utilisé par goldMEDAL regroupe ainsi les concepts de *dataset* avec l'ensemble de ses sous-classes (*dataset* interne, *dataset* externe, *dataset* structuré, etc.). De même, les groupements de goldMEDAL jouent le même rôle que les *tags* dans le modèle de Ravat et Zhao (2019a), c'est-à-dire catégoriser les données. Enfin, on retrouve les concepts de liens et de processus dans les deux modèles.

Toutefois, les concepts d'utilisateur et d'accès définis par Ravat et Zhao (2019a) ne trouvent pas de correspondance directe dans goldMEDAL. Les fonctions représentées par ces concepts peuvent cependant être implémentées à travers les processus.

La Table 6.2 illustre les correspondances entre les concepts de goldMEDAL et ceux de Ravat et Zhao (2019a).

Table 6.2 – Correspondance entre les concepts de goldMEDAL et du modèle de Ravat & Zhao

goldMEDAL	**Ravat & Zhao**
Entité de données	*Dataset* et sous-classes
Groupement	*Tag*
Lien	Lien
Processus	Processus
—	Utilisateur, Accès

Concepts de goldMEDAL *vs.* concepts de MEDAL

Les quatre concepts de goldMEDAL constituent également une généralisation des concepts de MEDAL. Dans goldMEDAL, les sous-classes associées au concept d'entité de données

dans MEDAL (entité de données originelle, version et représentation) sont omises. Seul le concept général d'entité de données est retenu. De façon analogue, les opérations de transformation et de mise à jour, ainsi que la relation de parenté définie dans MEDAL se traduisent dans goldMEDAL par l'unique concept de processus. Enfin, les concepts de groupement et de relation de similarité de MEDAL sont réutilisés dans goldMEDAL quasiment tels quels.

La Table 6.3 illustre les correspondances entre les concepts de goldMEDAL et ceux de MEDAL.

TABLE 6.3 – Correspondance entre les concepts de goldMEDAL et de MEDAL

goldMEDAL	**MEDAL**
Entité de données	Entité de données et sous-classes
Groupement	Groupement
Lien	Relation de similarité
Processus	Opérations de mise à jour et de transformation, Relation de parenté

Concepts de goldMEDAL *vs.* concepts de HANDLE

Le concept de processus n'a aucun équivalent dans HANDLE. Par conséquent, seuls les trois autres concepts de goldMEDAL correspondent à des concepts de HANDLE. Ainsi, HANDLE exploite les concepts de données brutes et de métadonnées pour représenter les données et les métadonnées atomiques qui leur sont associées, là où goldMEDAL utilise l'unique concept d'entité de données. De même, les concepts de catégorisation, de zone et de granularité dans HANDLE correspondent au groupement de goldMEDAL. Le concept de lien est quant à-lui commun aux deux modèles.

La Table 6.4 illustre les correspondances entre les concepts de goldMEDAL et de HANDLE.

TABLE 6.4 – Correspondance entre les concepts de goldMEDAL et de HANDLE

goldMEDAL	**HANDLE**
Entité de données	Données brutes, Metadonnées
Groupement	Categorisation, indicateur de zone indicateur de granularité
Lien	Lien
Processus	—

6.5 Conclusion

Dans ce chapitre, nous avons introduit goldMEDAL, une évolution de notre précédent modèle de métadonnées. goldMEDAL permet de modéliser les données et métadonnées d'un lac de données à travers quatre concepts que sont l'entité de données, le groupement, le lien et le processus. Au niveau logique, goldMEDAL exploite la théorie des graphes, comme MEDAL précédemment. Au niveau physique, goldMEDAL peut être implémenté avec un SGBD orienté graphes, ce que nous avons illustré à travers le cas d'usage du lac de données locatives HOUDAL (Scholly et al., 2021a).

L'analyse de goldMEDAL montre qu'il est le plus générique des modèles de métadonnées de la littérature, c'est-à-dire celui qui supporte la plus grande diversité de cas d'usage. Notre comparaison montre également que goldMEDAL est suffisamment abstrait pour représenter l'organisation des métadonnées à l'aide d'un nombre minimal de concepts. Ce haut niveau d'abstraction permet à goldMEDAL d'être simple à présenter et à percevoir.

Cependant, cette simplicité de compréhension pourrait en contrepartie engendrer des ambiguïté lors de la mise en œuvre du modèle. En effet, les concepts de goldMEDAL étant abstraits, leur traduction en métadonnées réelles lors de la mise en œuvre opérationnelle est déléguée aux concepteurs de lacs de données. Ceux-ci doivent alors faire appel soit à leurs propres connaissances, soit à des modèles de métadonnées plus directifs pour les guider dans l'identification effective des métadonnées. En ce sens, goldMEDAL peut être vu comme un complément des modèles plus directifs comme MEDAL.

Troisième partie

Contributions sur la mise en œuvre de lacs de données

Chapitre 7

Analyse assistée de documents textuels et tabulaires avec AUDAL

Sommaire

7.1 Introduction

Notre thèse se déroule dans le cadre du projet AURA-PMI et vise à remédier à une double problématique. D'une part, nous entendons répondre aux besoins d'analyse spécifiques au projet AURA-PMI à travers un lac de données adapté. D'autre part, nous visons de façon plus générale à remédier aux limites des approches existantes de modélisation et de mise en œuvre de lacs de données.

Pour répondre à cette double problématique (métier et scientifique), nous introduisons dans ce chapitre un lac de données dédié à l'analyse de données textuelles et tabulaires nommé AUDAL (*AURA-PMI DAta Lake*). Nous proposons une nouvelle approche méthodologique de conception et de mise en œuvre d'un lac de données. Ce faisant, nous nous focalisons particulièrement sur la problématique de gestion des métadonnées dont le rôle est crucial dans un lac de données. Grâce à une vision étendue des métadonnées, nous contribuons à l'avancement de la littérature relative aux lacs de données sur cinq aspects.

1. **Exploitation des lacs de données par des utilisateurs métiers**. La grande majorité des travaux sur les lacs de données réserve leur utilisation à des spécialistes du traitement de données (Fang, 2015 ; Khine & Wang, 2017 ; Madera & Laurent, 2016), ce qui exclut de fait les utilisateurs métiers. Ces derniers représentent pourtant une expertise précieuse dans le processus d'analyse des données. La nécessité d'ouvrir les lacs aux utilisateurs métiers est toutefois de plus en plus reconnue (Bagozi et al., 2019 ; Sulova, 2019), mais il existe encore peu d'approches permettant d'y parvenir. C'est une des problématiques auxquelles nous contribuons à répondre avec AUDAL.
2. **Intégration de données non structurées**. Les données non structurées représentent la majorité des données produites dans le monde (Miloslavskaya & Tolstoy, 2016). Pourtant, leur exploitation au sein des lacs de données n'est abordée que de façon marginale dans la littérature scientifique. En effet, à notre connaissance, seul le lac de données CODAL (*COREL*[1] *DAta Lake*) que nous avons proposé précédemment répond à cette problématique, à travers la prise en charge de documents textuels (Sawadogo et al., 2019a). Cependant, CODAL s'est révélé limité en termes de passage à l'échelle. AUDAL a pour objectif une amélioration substantielle de CODAL, dans tous ses aspects, pour la prise en charge de documents textuels dans les lacs de données.
3. **Intégration de données structurées**. La prise en charge de données structurées dans un lac de données peut sembler simple et évidente de prime abord. Cependant, dans un contexte de documents tabulaires bruts (fichiers CSV – *Comma Separated Values*), pouvoir interroger les données à l'aide d'un langage de requête (comme SQL) est un défi en soi. La simple prise en charge de langages de requêtes est d'ailleurs insuffisante pour rendre les données exploitables par des utilisateurs métiers. En guise de solution, AUDAL exploite un ensemble de métadonnées pour faciliter et assister l'interrogation des données structurées.

1. Le projet COREL (Au COeur de la RELation-clients) visait à analyser la politique d'un ensemble d'entreprises en matière de relation-clients, sur la base de leurs communications écrites

4. **Utilisation de technologies libres alternatives à Hadoop**. La plupart des approches d'implémentation de lacs de données (environ 75 %) se basent partiellement ou entièrement sur la technologie Apache Hadoop (Russom, 2017). Cependant, l'utilisation d'Hadoop nécessite une expertise dont les petites et moyennes entreprises (PME) ne disposent pas forcément. De plus, dans le contexte spécifique du projet AURA-PMI, les contraintes financières liées au projet empêchent le recours à des services de lacs de données dans le *cloud* comme Microsoft Azure ou Amazon AWS. C'est pourquoi AUDAL s'appuie exclusivement sur des outils libres. Nous entendons ainsi proposer une approche d'implémentation de lacs de données compatible non seulement avec le contexte des PME, mais aussi avec des projets de recherche dont les moyens peuvent être limités, tout en promouvant la science ouverte[2].
5. **Large panoplie d'analyses**. Dans la littérature, les propositions de lacs de données sont pour la plupart limitées à un type d'analyse. En effet, certains travaux se focalisent sur la tâche de recherche de données (Fernandez et al., 2018 ; A. Y. Halevy et al., 2016 ; Nargesian et al., 2018a), tandis que d'autres visent à permettre l'analyse du contenu des données (Bagozi et al., 2019 ; Hai et al., 2016 ; Malysiak-Mrozek et al., 2018). L'approche méthodologique que nous proposons vise à dépasser ce clivage en permettant à la fois la recherche de données et l'analyse de leurs contenus.

Le lac de données AUDAL a été accepté pour publication sous forme d'un article long en conférence internationale sous la référence Sawadogo, P. N., Darmont, J. & Noûs, C. (2021b). Joint Management and Analysis of Textual Documents and Tabular Data within the AUDAL Data Lake. *Proceedings of the 25th European Conference on Advances in Databases and Information Systems (ADBIS 2021), Tartu, Estonia*.

Dans la suite de ce chapitre, nous détaillons comment AUDAL répond aux exigences sus-mentionnées. Pour ce faire, nous présentons dans la Section 7.2 un aperçu du fonctionnement d'AUDAL. Dans la Section 7.3, nous détaillons son modèle de métadonnées aux niveaux logique et physique. Dans la Section 7.4, nous abordons la mise en œuvre de la couche d'accès aux données. Enfin, nous discutons et comparons les travaux connexes dans la Section 7.5.

7.2 Fonctionnement d'AUDAL

Dans cette section, nous présentons les principales caractéristiques de notre lac de données. Pour ce faire, nous montrons d'abord comment AUDAL se conforme à l'organisation en zones communément utilisée dans les lacs de données. Nous présentons ensuite l'architecture globale du système.

7.2.1 Mise en œuvre du polymorphisme des données

Les lacs de données suivent une approche *schema-on-read*, c'est-à-dire que les données ne sont pas transformées *a priori* (avant leur intégration), mais cela ne signifie pas qu'elles

2. https://www.enseignementsup-recherche.gouv.fr/pid39205/www.enseignementsup-recherche.gouv.fr/pid39205/science-ouverte.html

ne sont jamais transformées. En effet, il est indispensable de raffiner les données du lac pour les rendre compréhensibles et faciliter leur utilisation. Cela est particulièrement vrai pour les données non structurées, dont une représentation structurée est requise pour l'analyse (Diamantini et al., 2018). Dans notre approche, cette transformation des données est automatisée et systématisée à travers le concept de polymorphisme des données (Section 5.5.1).

Le polymorphisme des données consiste à créer et conserver simultanément plusieurs représentations brutes ou raffinées des même données dans le lac (Sawadogo et al., 2019b). Outre la nécessité de transformation des données brutes, ce principe est également motivé par le fait que chaque type d'analyse requiert un format spécifique de données. Par conséquent, prégénérer et conserver différentes représentations d'une même entité de données brute permet de prendre en charge et de faciliter une plus grande diversité d'analyses (Armbrust et al., 2021 ; Leclercq & Savonnet, 2018). La Figure 7.1 illustre le principe de polymorphisme des données à travers un exemple tiré des travaux de Leclercq et Savonnet (2018). Cet exemple montre qu'un corpus de documents textuels peut être représenté sous la forme d'une matrice documents-termes pour y appliquer des algorithmes d'analyse de texte (recherche des mots les plus fréquents, par exemple), ou encore sous la forme de graphe pour y appliquer des algorithmes de détection de communauté ou d'analyse de la centralité.

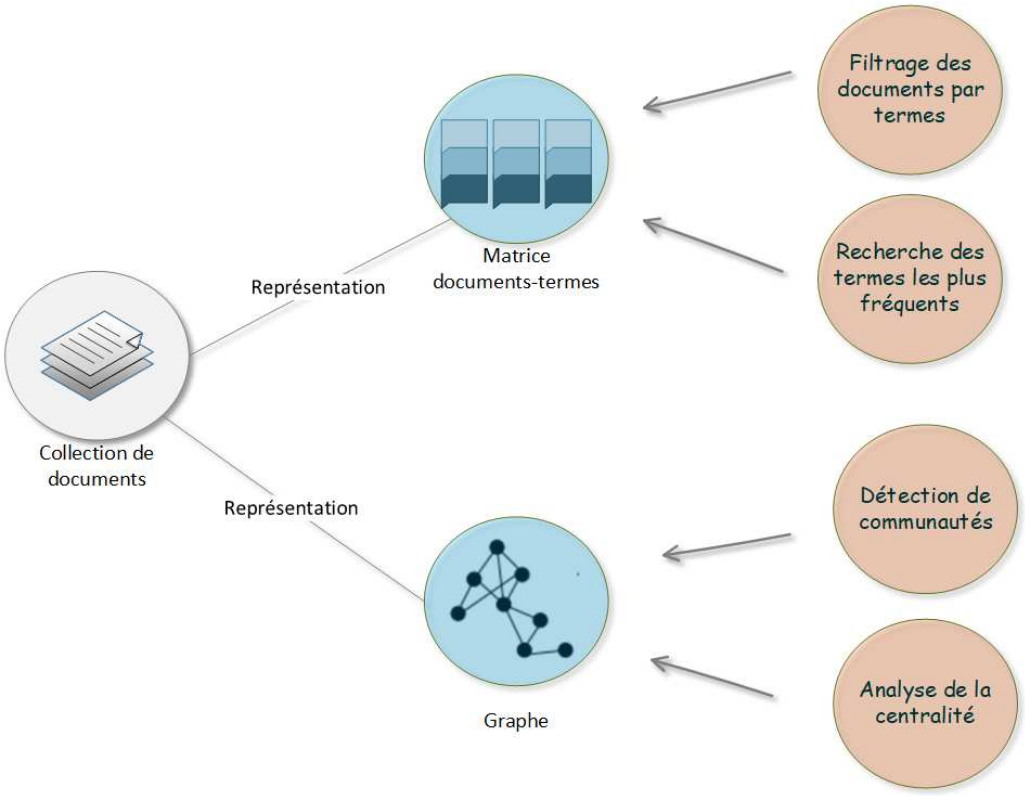


Figure 7.1 – Illustration du principe de polymorphisme des données

Ces représentations structurées peuvent être vues comme des métadonnées associées aux données brutes, puisqu'elles en facilitent l'analyse. Pour rappel, nous définissons les métadonnées comme des « informations structurées qui décrivent, expliquent, localisent, ou

encore facilitent ou permettent la recherche, l'utilisation ou la gestion d'une ressource d'information » (Riley, 2017). Nous implémentons donc le polymorphisme à travers un processus automatisé de génération de métadonnées. Plus concrètement, nous considérons chaque entité de données (initiale) comme étant une représentation brute. Ensuite, un script permet de parcourir l'ensemble de ces représentations brutes (de documents textuels ou tabulaires) pour en générer des représentations raffinées. Ces nouvelles représentations sont stockées sous forme de métadonnées.

Cette approche peut être comparée à l'organisation classique de lacs de données en zones. Pour rappel, l'organisation en zones désigne le fait de catégoriser les données du lac selon leur niveau de raffinement (Section 4.2.1). L'organisation en zones vise avant tout à séparer les données brutes, difficilement exploitables des données plus raffinées prêtes à être consommées.

Il faut toutefois noter deux différences entre l'approche en zones classique et la nôtre. La première différence concerne le sens donné aux données transformées. Dans notre approche, les représentations sont traitées comme des données auxiliaires associées aux entités de données dont elles sont issues. Ainsi, nous restons centrés sur l'entité de données originelle, les représentations ne servant qu'à en faciliter analyse. Cela permet de mieux tracer les relations généalogiques entre les données brutes et les données raffinées. *A contrario*, les approches en zones traitent généralement les données raffinées indépendamment des données brutes dont elles sont issues, ou suppriment même les donnés brutes (B. Inmon, 2016 ; LaPlante & Sharma, 2016).

La seconde différence entre notre approche par polymorphisme des données et les approches classiques en zones concerne l'automatisation du processus de raffinement. Dans notre approche, des transformations sont appliquées à plusieurs entités de données, voire à toutes les entités de données de même type. Cela permet d'automatiser cette tâche. Cette automatisation est en revanche difficilement réalisable dans les approches en zones, où les entités de données sont transformées de façon ponctuelle, suivant des procédés ad hoc.

En somme, le principe de polymorphisme des données d'AUDAL peut être résumé comme un moyen d'étendre et d'accélérer les analyses, tout en respectant l'approche *schema-on-read* propre aux lacs de données.

7.2.2 Architecture d'AUDAL

Nous définissons l'architecture du lac de données AUDAL à travers une organisation fonctionnelle en trois couches associées chacune à une fonction du lac de données : une couche de stockage des données et métadonnées, une couche de gestion des métadonnées et une couche d'accès aux données (Figure 7.2).

1. La **couche de stockage** est chargée de stocker les données brutes, ainsi que les métadonnées (y compris les données raffinées). Pour ce faire, elle exploite une combinaison de systèmes de stockage, chacun étant adapté à un besoin de stockage spécifique. Ainsi, nous utilisons un simple système de fichiers pour le stockage des données brutes, un SGBD orienté graphes pour matérialiser les relations entre les

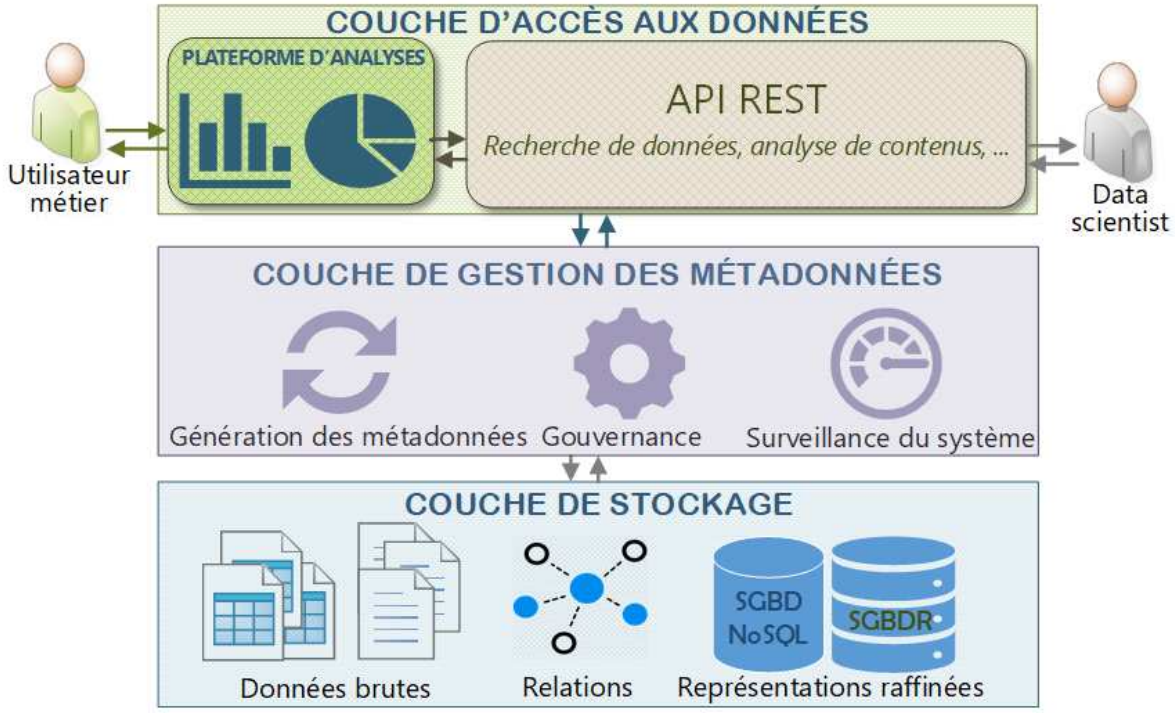


FIGURE 7.2 – Architecture fonctionnelle d'AUDAL

entités de données, un SGBD orienté documents pour le stockage des représentations raffinées de documents textuels et un SGBD relationnel pour le stockage des représentations raffinées de documents tabulaires.

2. La **couche de gestion des métadonnées** est constituée d'un ensemble de processus dédiés à l'extraction et à la génération automatique des métadonnées. C'est à travers cette couche que le principe de polymorphisme des données est appliqué. Ce faisant, elle facilite indirectement l'exploitation des données, évitant ainsi le syndrome de marécage de données, qui est souvent causé par l'absence et l'insuffisance de politique de gestion des métadonnées. Cette couche permet en outre de surveiller les infrastructures logicielles qui prennent en charge les couches de stockage et d'accès.

3. La **couche d'accès aux données** permet enfin d'interroger les données stockées dans AUDAL. Elle est constituée de deux interfaces complémentaires. D'une part, une API REST permet d'analyser ou d'extraire les données du lac à l'aide d'un formalisme précis. D'autre part, une interface graphique permet de réaliser les mêmes traitements à travers une application web. Ces deux outils permettent un accès différencié aux données du lac en fonction du profil des utilisateurs. Ainsi, les *data scientists* accèdent aux données du lac à travers l'API REST et les utilisateurs métiers via l'interface graphique. Cela confère à AUDAL un caractère inclusif, *a contrario* d'une vision largement répandue des lacs de données qui exclut les utilisateurs métiers (FANG, 2015 ; KHINE & WANG, 2017 ; MADERA & LAURENT, 2016).

7.3 Stockage et organisation des métadonnées

Dans cette section, nous présentons notre approche de gestion des métadonnées en trois étapes. D'abord, nous identifions et organisons les concepts à intégrer dans le système de métadonnées à travers des modèles conceptuel et logique, respectivement. Nous montrons ensuite comment le modèle logique peut être traduit en modèle physique. Enfin, nous détaillons les méthodes et technologies utilisées pour la génération et l'extraction des métadonnées.

7.3.1 Modélisation conceptuelle et logique

Dans cette section, nous présentons notre approche d'organisation des métadonnées. Pour ce faire, nous identifions d'abord les métadonnées pertinentes pour notre cas d'usage, que nous organisons ensuite en modèles conceptuel et logique.

Identification des métadonnées

Le système de métadonnées que nous proposons est basé sur le modèle MEDAL. Pour rappel, le choix de MEDAL s'est imposé du fait que le modèle goldMEDAL n'existait pas encore lors de la conception du systeme AUDAL. Nous avons ainsi été guidé par MEDAL pour le choix des métadonnées à intégrer dans AUDAL. Tout en restant fidèle aux spécifications de MEDAL, nous avons toutefois ignoré certaines métadonnées qui sont inadaptées à notre cas d'usage. Ainsi, nous retenons les métadonnées suivantes.

Métadonnées intra-entité Pour rappel, ce sont les métadonnées directement associées aux entités de données brutes. Deux types de métadonnées intra-entité sont intégrés dans AUDAL.

Les *propriétés* sont des informations, le plus souvent atomiques qui caractérisent une entité de données. Dans AUDAL, nous considérons au titre de ces métadonnées le nom de l'auteur du document, l'URL de stockage de l'entité de données dans le système de fichier, le titre du fichier, ainsi que l'horodatage de création et de dernière modification. Dans le cas spécifique des documents tabulaires, nous ajoutons des propriétés plus complexes qui décrivent les colonnes contenues dans chaque document. Ces métadonnées particulières définissent pour chaque colonne le type de contenu (nombres, textes ou dates), la proportion de valeurs différentes (rapport entre nombre de valeurs différentes et le nombre de lignes de la table), et la proportion de valeurs nulles. Nous notons également pour les documents textuels le nombre de n-uplets et de colonnes.

Les *représentations raffinées* sont des entités de données structurées de sorte à permettre une analyse automatique des entités de données brutes dont elles sont issues. Dans AUDAL, nous considérons deux types de représentations raffinées pour les documents textuels. Une représentation en sac de mots permet de retrouver les mots les plus fréquents dans un ou plusieurs documents. Toutefois, cette représentation sied peu aux calculs de distances entre documents, du fait de sa grande dimensionnalité (chaque terme est une dimension potentielle). Dans le but de prendre en charge des analyses impliquant des calculs de distance entre les documents, nous avons également défini pour chaque document une

représentation de type *embedding*. Ce type de représentation traduit en effet chaque document textuel en un vecteur de seulement quelques dizaines de dimensions, ce qui le rend particulièrement adapté aux calculs de distance. Dans le cas des documents tabulaires, nous considérons une seule et unique représentation raffinée qui consiste à transformer l'entité de données brute en une table relationnelle.

Métadonnées inter-entités Ces métadonnées traduisent les connexions entre les entités de données du lac. Nous retenons deux types de métadonnées inter-entités dans AUDAL.

Les *groupements* permettent d'organiser les entités de données en collections. Ils peuvent théoriquement regrouper indistinctement des documents tabulaires et textuels sur la base de caractéristiques communes. Cependant, dans le contexte du projet AURA-PMI, l'absence de caractéristiques communes aux documents tabulaires et textuels, combinée au faible nombre de documents tabulaires dans les données initiales (seulement 6), nous a conduit à créer des groupements associés exclusivement aux documents textuels. Nous avons ainsi défini pour les données du projet AURA-PMI, des groupements basés sur la langue (français et anglais), le type MIME du document, la catégorie du document, l'entreprise à laquelle le document est associé, l'année et le mois de création du document, ainsi que le secteur d'activité, le sous-secteur d'activité et la région de l'entreprise à laquelle le document est associé. Notons que des groupements peuvent être ajoutés, modifiés ou supprimés à la demande lors de la mise à jour des processus de génération des métadonnées.

Les *relations de similarité* mesurent quant à elles la ressemblance de contenu entre des paires d'entités de données. Dans le cas des documents textuels, nous définissons ces métadonnées entre les représentations raffinées à l'aide de la similarité cosinus, qui est une mesure de similarité adaptée aux textes. Pour les données tabulaires, nous utilisons comme relation de similarité les dépendances fonctionnelles de type clé primaire-clé étrangère entre attributs. En reliant ces colonnes, nous connectons ainsi les documents tabulaires correspondants.

Métadonnées globales Ces métadonnées sont des structures de données construites et enrichies en permanence pour faciliter et optimiser les analyses dans le lac. AUDAL comprend deux types de métadonnées globales.

Les *ressources sémantiques* sont des bases de connaissances qui permettent d'améliorer à la fois la génération de métadonnées et la recherche de données. Dans AUDAL, des dictionnaires servent ainsi à la génération de représentations raffinées en sacs de mots pour les documents textuels. Les thésaurus servent quant à eux à étendre automatiquement les recherches par mots clés avec des synonymes.

Les *index inversés* permettent d'établir une correspondance entre des termes et les entités de données dans le lac. Ils sont particulièrement utiles pour prendre en charge et accélérer les recherches par mots clés.

Modèle conceptuel

La phase précédente d'identification des métadonnées nous a permis de déterminer les métadonnées appropriées pour la conception de notre système de métadonnées. Nous avons ainsi retenu les propriétés, les représentations, les groupements, les relations de similarité, les ressources sémantiques et les index inversés. Toutes ces métadonnées constituent une sous-partie du modèle conceptuel de MEDAL que nous présentons, à l'exception des métadonnées globales qui sont gérées à part, dans la Figure 7.3.

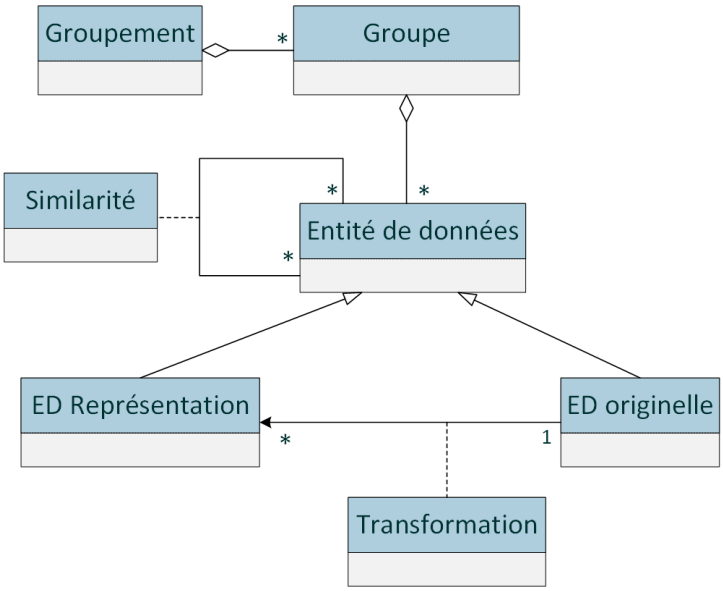


FIGURE 7.3 – Modèle conceptuel du système de métadonnées d'AUDAL

Modèle logique

Conformément au modèle MEDAL, la modélisation logique des métadonnées dans AUDAL se fait à l'aide d'éléments de la théorie des graphes. Les Figures 7.4 et 7.5 illustrent la modélisation logique des métadonnées intra-entité relatives aux documents textuels et tabulaires, respectivement. Chacune de ces figures présente une entité de données originelle à laquelle on associe une ou plusieurs représentations.

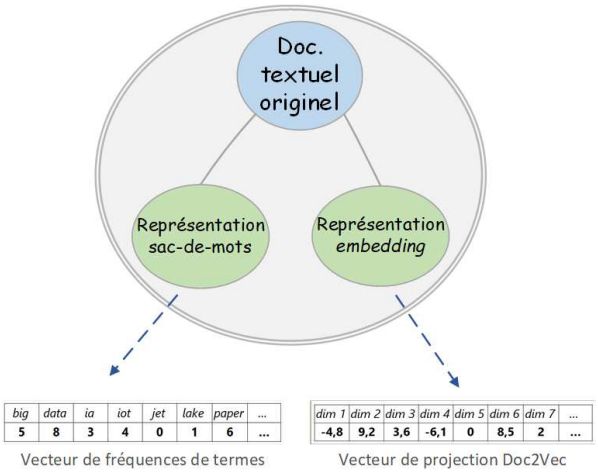


FIGURE 7.4 – Exemple de modèle logique de métadonnées intra-entité de documents textuels

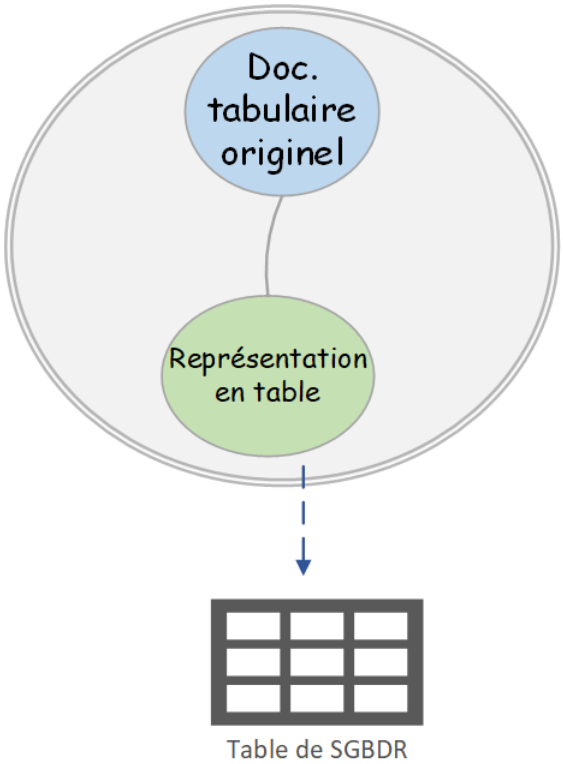


FIGURE 7.5 – Exemple de modèle logique de métadonnées intra-entité de documents tabulaires

Les Figures 7.6 et 7.7 présentent quant à elles l'organisation logique des relations de similarité. On remarque que ces relations sont orientées dans le cas de documents tabulaires, alors qu'elles ne le sont pas pour les documents textuels.

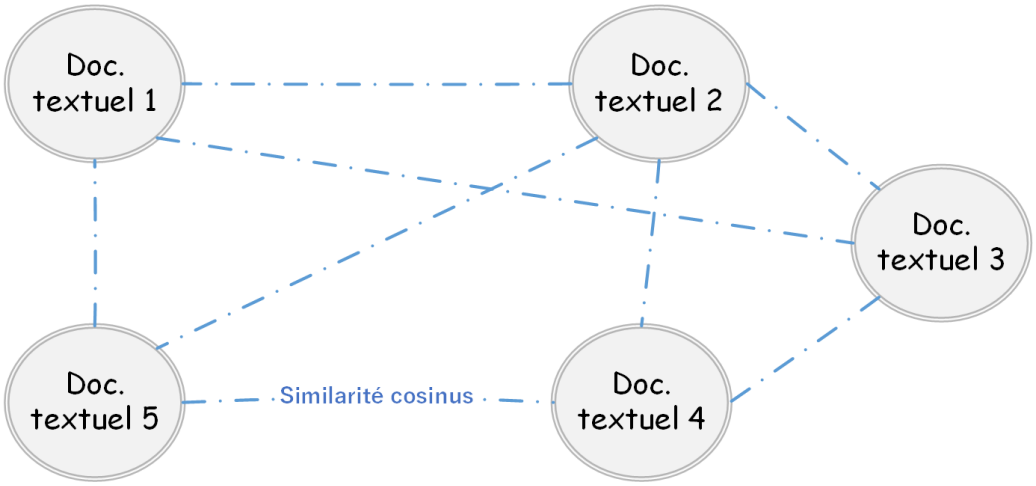


FIGURE 7.6 – Exemple de modèle logique de relations de similarité entre documents textuels

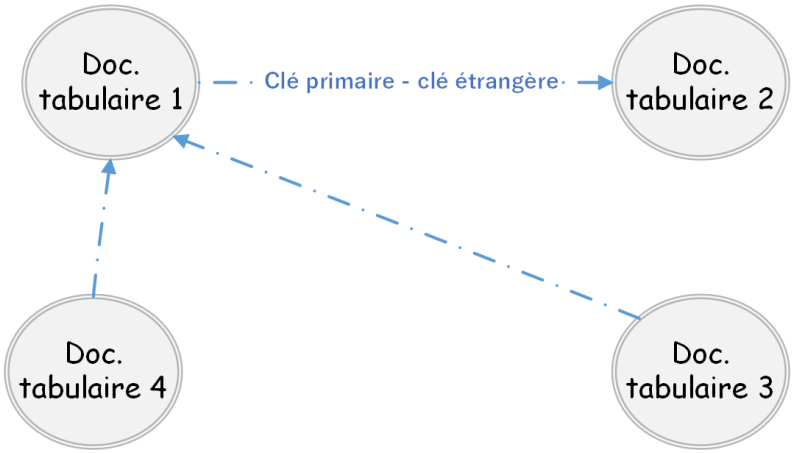


FIGURE 7.7 – Exemple de modèle logique de relations de similarité entre documents tabulaires

7.3.2 Modélisation physique et approche de stockage

La modélisation physique consiste à adapter le modèle logique aux technologies de stockage utilisées en pratique. C'est pourquoi nous présentons et motivons notre approche de stockage, avant de présenter le modèle physique et la façon dont il est implémenté.

Technologies de stockage

Le modèle logique que nous proposons suit une approche par graphes. Son implémentation se fait donc de façon naturelle via un SGBD orienté graphes. Bien que le formalisme RDF et d'autres technologies du Web sémantique constituent des alternatives pour l'expression de relations entre des entités de données, ces approches font face à une problématique de passage à l'échelle (AKHTER et al., 2018). C'est pourquoi nous basons l'implémentation de notre modèle de métadonnées sur le **SGBD orienté graphes Neo4J**. Ce dernier a en effet l'avantage de passer à l'échelle grâce à un système de réplication des données sur

un *cluster*.

Nous avons associé Neo4J à d'autres SGBD, pour des besoins de stockage spécifiques qu'il ne prend pas en charge nativement. Nous avons sélectionné ces solutions complémentaires sur la base non seulement des fonctions proposées, mais aussi par rapport à leurs capacités de passage à l'échelle et en fonction des types de licences d'utilisation.

— Le **SGBD orienté documents MongoDB** sert à stocker les représentations raffinées de documents textuels, c'est-à-dire, des sacs de mots et de vecteurs d'*embeddings*. Chaque représentation est stockée dans MongoDB sous la forme d'une liste de couples clés-valeurs. MongoDB a le double avantage d'offrir une licence libre et de fournir un système de distribution qui permet de passer à l'échelle.

— Le **système d'indexation Elasticsearch** est utilisé pour stocker et maintenir les index inversés. Comme MongoDB et Neo4J, Elasticsearch propose une licence libre, et intègre un système de distribution qui lui confère une capacité de passage à l'échelle.

— Le **SGBD relationnel SQLite** permet enfin de stocker les représentations raffinées de documents tabulaires, qui sont purement et simplement des tables relationnelles. Le choix de ce système s'explique par sa légèreté et sa faible consommation en ressources.

Modèle physique

Les Figures 7.8, 7.9, 7.10 et 7.11 représentent le modèle physique de AUDAL à travers son implémentation sous Neo4J, qui est le cœur du système. Ce modèle physique montre que les représentations raffinées sont virtuellement présentes dans Neo4J sous forme de nœuds, bien qu'elles soient physiquement stockées dans des systèmes annexes. On constate également que les descriptions associées aux colonnes de documents tabulaires donnent lieu à des nœuds. Cela s'explique par le fait que Neo4J ne supporte pas d'attributs complexes à l'intérieur des nœuds. Par conséquent, il est nécessaire d'extérioriser ces descriptions sous formes de nœuds spéciaux auxquels nous avons attribué le label COLUMN.

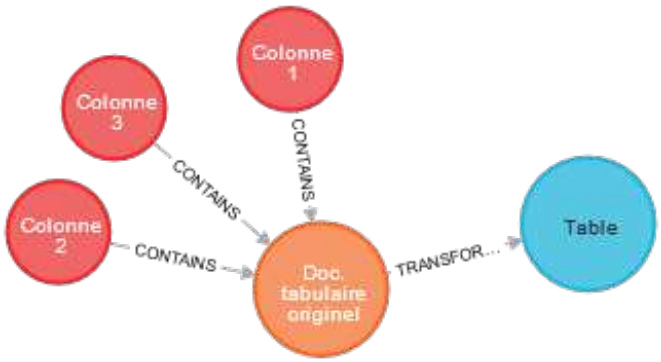


FIGURE 7.8 – Exemple de modèle physique de métadonnées intra-entité de documents tabulaires

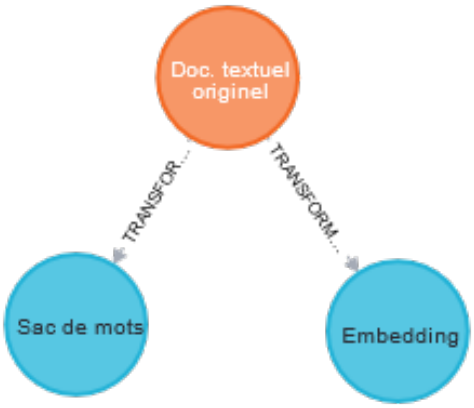


FIGURE 7.9 – Exemple de modèle physique de métadonnées intra-entité de documents textuels

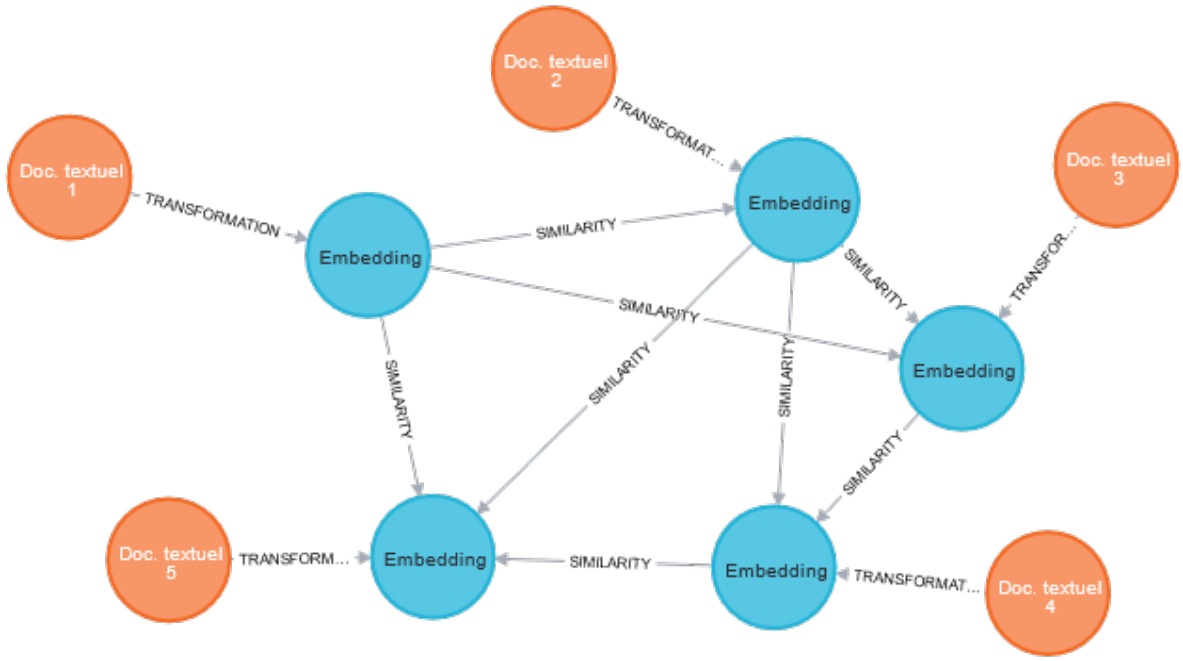


FIGURE 7.10 – Exemple de modèle physique de relations de similarité entre documents textuels

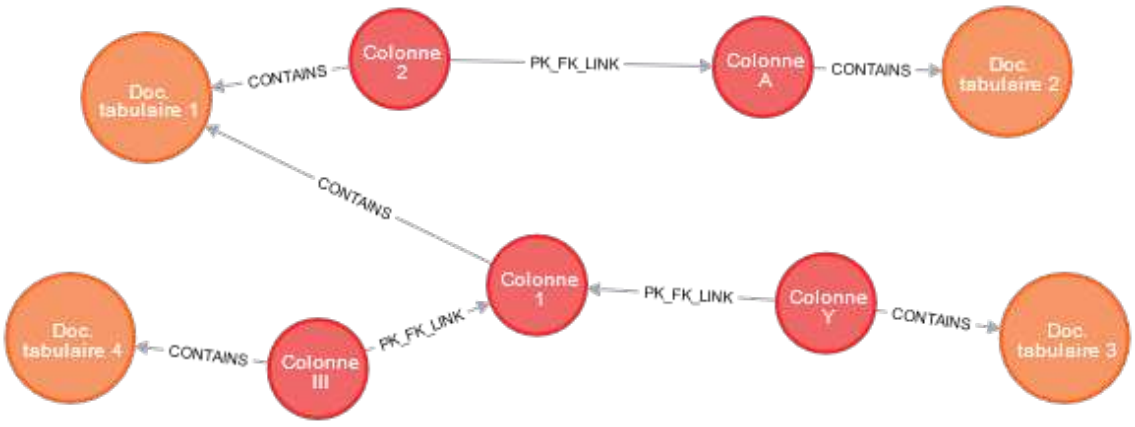


FIGURE 7.11 – Exemple de modèle physique de relations de similarité entre documents tabulaires

Déploiement de la couche de stockage sur l'infrastructure physique

AUDAL a été implémenté sur un cluster de trois machines virtuelles VMware[3] sous le système d'exploitation Ubuntu 18.04 (Table 7.1). La première machine virtuelle est dotée d'un processeur Intel-Xeon à 7 cœurs cadencés à 2,20 GHz et d'une mémoire vive de 24 Go. Les deux autres machines virtuelles ont chacune un processeur mono-core Intel-Xeon à 2.20 GHz ainsi que 24 Go de RAM.

Sur cette infrastructure, nous avons déployé des clusters de trois instances pour chacun des systèmes Neo4J, Mongo DB et Elasticsearch. Chaque machine virtuelle supporte ainsi en parallèle trois différents systèmes de stockage. De façon plus spécifique, la machine n° 1 supporte en plus le SGBD SQLite, le stockage des données brutes à travers son système de fichiers, ainsi que les scripts de génération des métadonnées.

TABLE 7.1 – Description du *cluster* d'AUDAL

Machine → **Critère ↓**	**Machine #1**	**Machine #2**	**Machine #3**
Support physique	VMware	VMware	VMware
Système d'exploitation	Ubuntu 18.04	Ubuntu 18.04	Ubuntu 18.04
Processeur	Intel Xeon	Intel Xeon	Intel Xeon
Processeur - fréquence	2.20 GHz	2.20 GHz	2.20 GHz
Processeur - cœurs	7 cœurs	mono-core	mono-core
RAM	24 Go	24 Go	24 Go
Disque dur	200 Go	100 Go	100 Go

7.3.3 Extraction et génération des métadonnées

Dans cette section, nous faisons le tour des technologies et techniques que nous avons utilisées dans le processus d'extraction ou de génération de métadonnées.

Extraction des propriétés

Certaines propriétés comme l'URL de stockage, la taille ou encore la date de création sont souvent encapsulées dans le document lui-même. Elles peuvent donc être obtenues par n'importe quel langage de programmation. Par contre, d'autres propriétés nécessitent des outils plus spécialisés pour être extraites ou détectées. C'est le cas de la langue dans laquelle un document est rédigé ou encore du type MIME. Nous extrayons ces métadonnées à l'aide de la bibliothèque Apache Tika. Nous l'utilisons en conjonction avec l'outil de reconnaissance optique de caractères Tesseract OCR[4] pour le cas de documents textuels numérisés. Ces documents textuels sont en réalité des images à partir desquelles les caractères doivent être extraits par des méthodes et outils spécifiques.

3. https://www.vmware.com/fr
4. https://github.com/tesseract-ocr/tesseract

Génération des représentations raffinées

Pour générer les deux types de représentations raffinées (sac de mots et *embedding*) à partir des documents textuels, nous appliquons des méthodes classiquement utilisées dans le domaine du traitement du langage naturel pour la préparation des données. En conjonction de ces deux types de représentations, nous avons conçu deux stratégies de raffinement sur la base du vocabulaire utilisé. L'une prend en compte un vocabulaire libre, tandis que l'autre se base sur un vocabulaire personnalisé et restreint. Le croisement de ces deux stratégies avec les deux types de représentations initialement prévus donnent lieu à quatre instances de représentations qui sont générées par un script Python.

1. **Sac de mots × vocabulaire libre.** Nous procédons d'abord à un chargement des documents textuels dans une liste. Chaque document est ensuite filtré des mots vides (*stop words*), c'est-à-dire les termes qui ne portent pas de sens en eux-mêmes (les articles, par exemple). Après cela, un processus de lemmatisation permet de ramener chaque terme à sa forme neutre canonique, à l'aide du *framework* TreeTagger[5]. Les textes résultant de cette étape de lemmatisation sont alors transformés en listes de couples clés-valeurs, dont chaque clé représente un terme et chaque valeur correspond au nombre d'occurrences de ce terme dans un document. Nous avons limité les termes considérés dans ce processus aux 2000 mots les plus fréquents dans l'ensemble des documents (hors mots vides).
2. **Sac de mots × vocabulaire personnalisé.** Ici, nous filtrons les documents non pas pour en exclure les mots vides, mais pour retenir uniquement des termes d'intérêt. Il s'agit plus précisément de mots qui traduisent la notion de digitalisation (données, informatique, numérique, automatique, logiciel, etc.) fournis par l'équipe « Sciences de gestion » du projet AURA-PMI. Les textes résultants sont ensuite lemmatisés et transformés en listes de couples clés-valeurs comme précédemment, mais en conservant cette fois l'intégralité du vocabulaire.
3. ***Embedding* × vocabulaire libre.** Pour obtenir ce type de représentation, nous procédons comme précédemment à un filtrage des mots vides, puis à une étape de lemmatisation. Nous appliquons ensuite la méthode Doc2Vec (Le & Mikolov, 2014), qui permet de projeter chaque document dans un espace vectoriel de dimensions réduites. Dans notre cas, nous avons utilisé 100 dimensions pour représenter chaque document. La représentation ainsi obtenue pour chaque document est alors transformée en liste de clés-valeurs, où les clés représentent les 100 dimensions (dim_1, dim_2, etc.), tandis que les valeurs traduisent les coordonnées du document sur ces dimensions.
4. ***Embedding* × vocabulaire personnalisé.** Il s'agit exactement du même processus que dans le cas précédent, sauf que les documents sont précédemment filtrés sur la base d'un vocabulaire fourni.

Les représentations raffinées de documents tabulaires sont quant à elles générées à l'aide des bibliothèques Python Pandas[6] et SqlAlchemy[7]. Ce processus consiste simplement à charger chaque document sous la forme d'un *dataframe*, pour ensuite l'exporter vers le SGBD SQLite sous la forme d'une table relationnelle.

5. https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/
6. https://pandas.pydata.org/
7. https://www.sqlalchemy.org/

Génération des groupements

Dans AUDAL, nous utilisons trois stratégies de création de groupements entre les documents. Dans la première stratégie, nous générons les différents groupes à partir de propriétés et leur associons les documents concernés. C'est typiquement le cas des groupements sur la langue, l'année, le mois ou encore le type MIME. Dans la seconde stratégie, nous déduisons les groupes à partir de l'URL de stockage. En effet, dans certains cas comme le nôtre, les répertoires situés au même niveau d'arborescence traduisent un groupement. C'est de cette façon que nous déduisons les groupements sur le type de document (article de presse, rapport d'activité, rapport financier, etc.) et l'entreprise concernée. Enfin, la troisième stratégie de détection de groupements consiste à généraliser un groupement pré-existant à travers une structuration hiérarchique. Plus concrètement, nous utilisons des informations ad hoc sur les entreprises pour étendre le groupement des documents par entreprise. Nous obtenons ainsi un groupement selon le secteur et le sous-secteur d'activité de l'entreprise à laquelle appartient le document, un autre selon la région où se situe le siège de l'entreprise, et ainsi de suite.

Génération des relations de similarité

Les relations de similarité entre documents textuels sont obtenues en mesurant la similarité cosinus (Allan et al., 2000) entre chaque couple de documents. La similarité cosinus est une mesure statistique communément utilisée en recherche d'information pour traduire la ressemblance entre des documents textuels.

Nous calculons la similarité cosinus entre chaque paire de documents en utilisant des représentations des documents sous forme d'*embeddings* Doc2Vec. Nous obtenons en résultat une matrice carrée de similarités. Pour réduire la complexité de ce processus, nous limitons le calcul aux valeurs situées en dessous de la diagonale de la matrice de similarités. Nous réduisons ainsi de moitié le temps de calcul. Pour faciliter et assurer le passage à l'échelle du stockage des mesures de similarités, nous introduisons une limitation supplémentaire aux mesures de similarité stockées. Ainsi, nous ne stockons pour chaque document que la similarité entre le document considéré et les cinq documents qui lui sont le plus proches. Cela permet de prendre en charge la recherche de documents connexes (Section 7.4.2) tout en assurant une croissance soutenable de l'infrastructure de stockage.

Concernant les documents tabulaires, nous générons les relations de type clé primaire-clé étrangère entre les colonnes en exploitant l'approche de Chen et al. (2015), qui est éprouvée et a déjà été utilisée dans le contexte des lacs de données (Fernandez et al., 2018).

Considérons $L.x$ et $R.y$ deux colonnes issues des tables L et R, respectivement. Un lien $COLUMN\_JOINABILITY$ est créé de $L.x$ à $R.y$, c'est-à-dire que $L.x$ est une clé étrangère référençant la clé primaire $R.y$ si et seulement si :

1. *$L.x$ et $R.y$ sont tous deux non numériques.* On part du principe que les clés primaires et étrangères sont plus susceptibles d'être non numériques ;
2. *le ratio de valeurs uniques dans $R.y$ est de 1.* Cela traduit le fait qu'une clé primaire doit être unique ;

3. *toutes les valeurs de L.x sont contenues dans R.x*. Une clé primaire doit contenir toutes les valeurs de sa clé étrangère.

Comme toutes les informations nécessaires à l'application des première et deuxième règles sont inclues dans le système de métadonnées (sous forme de descriptions de colonnes), nous optimisons le processus de génération de ces relations en filtrant *a priori* les colonnes qui répondent aux critères ci-dessus. Cela permet de réduire le nombre de combinaisons à vérifier, et donc le temps nécessaire à l'exécution du processus.

Indexation

Tous les documents (textuels et tabulaires) sont indexés à l'aide d'un script Java qui fait appel à une bibliothèque d'interaction avec Elasticsearch. Les deux types de documents sont stockés dans des index séparés, mais un alias permet de les interroger simultanément. Pour les documents tabulaires, seules les valeurs textuelles contenues dans les colonnes sont indexées. Les documents textuels sont quant à eux indexés intégralement. Le script d'indexation, ainsi que ceux qui servent à la génération des relations de similarité, intègrent un système d'exécution parallèle qui permet de réduire de façon conséquente la durée des traitements.

L'ensemble des scripts utilisés pour la génération des métadonnées est disponible en accès ouvert [8].

7.4 Couche d'accès aux données

Dans cette section, nous détaillons le fonctionnement de la couche d'accès aux données en présentant son architecture, les technologies utilisées, ainsi que les fonctions proposées.

7.4.1 Architecture de la couche d'accès

La double interface d'accès aux données prend la forme d'une application web, c'est-à-dire, une application client-serveur accessible via un navigateur web. Elle suit une architecture *back-end-front-end*. Cette structure offre une modularité qui facilite l'évolutivité et la maintenance du système. Elle nous permet en outre de fournir un accès différencié aux données du lac, selon les compétences de l'utilisateur.

La partie ***back-end*** interagit directement avec la couche de stockage, à partir de laquelle elle extrait et rend disponible les données à travers des micro-services. Ces micro-services sont des fonctions prédéfinies (mais paramétrables et extensibles) qui facilitent et uniformisent l'interrogation des données et métadonnées du lac (Section 7.4.2). Le *back-end* d'AUDAL est basé sur le *framework* Java Spring [9].

La partie ***front-end*** est en pratique une interface graphique qui donne un accès intuitif et visuel aux données du lac. Elle réutilise les micro-services développés en *back-end*. Des

8. https://github.com/Pegdwende44/AudalMetadata
9. https://spring.io/

captures d'écran de l'interface d'analyse sont présentées en Annexe A. Sur le plan technologique, la partie *front-end* est basée sur le *framework* Javascript AngularJS [10].

Le code source des deux composants de la couche d'analyse est disponible en ligne [11].

7.4.2 Analyses avec AUDAL

La double interface d'accès aux données d'AUDAL permet aux utilisateurs d'extraire les données du lac sous leurs formes brutes, ou alors de réaliser un ensemble d'analyses à l'aide de fonctions prédéfinies mais extensibles, présentées à travers des services. Ces services peuvent être regroupés en deux types : les services de recherche de données, d'une part, et les services d'agrégation ou d'analyse de contenu, d'autre part. Les services de recherche de données implémentés dans AUDAL sont transversaux, c'est-à-dire qu'ils sont adaptés à la fois aux documents tabulaires et textuels. Ils contrastent ainsi avec les services d'agrégation et d'analyse du contenu, qui sont spécifiques à chaque type de documents.

7.4.3 Services de recherche de données

La recherche de données consiste à retrouver des documents stockés dans le lac, sur la base de caractéristiques précises. Ces services, qui sont au nombre de trois, peuvent être utilisés séparément ou conjointement.

1. Le service de **recherche par mots clés** consiste à filtrer les données selon qu'elles contiennent (ou non) un ensemble de mots clés définis par l'utilisateur. Dans AUDAL, une option de « recherche floue » permet d'étendre la requête en considérant des termes syntaxiquement proches de ceux définis par l'utilisateur. Une autre option permet d'étendre la requête avec des termes sémantiquement similaires sur la base d'un thésaurus.
2. Le service de **navigation** exploite les groupements pour permettre de retrouver des entités de données en croisant plusieurs groupes. Par exemple, on peut retrouver les documents associés à une entreprise précise et édités à une année donnée en croisant les groupes « Entreprise X » et « 2010 » issus des groupements par entreprise et par année, respectivement.
3. Le service de **recherche de données connexes** retrouve des documents sur la base de leur proximité avec un document précis. Cette proximité est établie à partir des relations de similarité qui sont stockées dans le système de métadonnées. Plus concrètement, dans le cas des documents tabulaires, les documents connexes sont des entités de données qui partagent une dépendance fonctionnelle de type clé primaire-clé étrangère avec l'entité de données d'intérêt. Dans le cas des documents textuels, il s'agit de données dont la similarité cosinus avec l'entité de données d'intérêt est relativement forte.

10. https://angularjs.org/
11. https://github.com/Pegdwende44/AUDAL

7.4.4 Services d'agrégation et d'analyse de contenus textuels

Les services portant sur le contenu des documents permettent d'extraire des informations à partir d'un ou plusieurs documents cibles. Dans le cas des documents textuels, l'utilisation de ces services ouvre la porte à des analyses de type OLAP (Codd et al., 1993). Les groupements peuvent en effet servir de dimensions en filtrant les données de multiples façons. Ainsi, les données du lac peuvent être rapidement et intuitivement réduites à un sous-ensemble par l'intersection de groupes, ce qui est comparable aux opérations OLAP *Slice & Dice*. Une fois les documents filtrés, ils peuvent être agrégés pour obtenir des informations raffinées. Les résultats agrégés peuvent être comparés entre différents sous-ensembles de documents. Dans l'interface graphique, les résultats agrégés sont traduits sous forme de visualisations. Les utilisateurs peuvent ainsi utiliser les services suivants pour extraire ou visualiser des informations d'un sous-ensemble de documents.

4. Le service ***top keywords*** résume les documents par une liste des mots clés les plus fréquents, en agrégeant des représentations en sacs de mots des documents. Grâce au principe de polymorphisme des données, les utilisateurs peuvent choisir entre plusieurs stratégies de définition des mots clés au moment d'extraire les *top keywords* (vocabulaire libre ou vocabulaire personnalisé). Dans l'interface graphique, les mots les plus fréquents sont présentés à travers un diagramme en barre ou un nuage de mots.
5. Le service de ***scoring*** consiste à évaluer numériquement la correspondance entre des documents et un ensemble de termes entrés par l'utilisateur. Parmi les techniques permettant d'attribuer un score aux documents, nous avons choisi l'algorithme d'Elasticsearch (2019), qui prend en compte les fréquences d'apparition des termes de requête dans les documents.

   Pour mieux expliciter le fonctionnement de cet algorithme, nous prenons le cas simple d'un *scoring* par rapport à un seul terme de requête. Dans ce cas, le score d'un document est le produit de trois valeurs définies et calculées comme suit.

   — La *fréquence du terme (TF)* représente le nombre d'occurrences du terme (t) de requête dans le document cible (d). Sa formule est :
   $TF(t\ dans\ d) = \sqrt{freq(t/d)}$.

   — La *fréquence inverse du terme dans les documents (IDF)* représente les apparitions du terme de requête dans l'ensemble des documents. Sa formule est :
   $IDF(t) = 1 + log(nb\_documents/(freq(t) + 1))$.

   — La *longueur normalisée du document* reflète le nombre total de termes différents dans le document cible. Sa formule est :
   $norm(d) = 1/\sqrt{nbTerms}$.

   Dans le cas des requêtes multi-termes, une formule similaire permet de générer un score de correspondance entre le document et l'ensemble des termes (Elasticsearch, 2019).

   En raison du grand nombre de documents, les scores individuels par document sont difficiles à lire et à interpréter. C'est pourquoi nous proposons dans l'interface graphique une visualisation des scores agrégés par groupe. Cela permet par exemple de comparer le score moyen des documents d'une entreprise au score moyen des documents d'une autre entreprise.

6. Le service de ***highlights*** permet d'obtenir des fragments de documents textuels où des termes définis par l'utilisateur apparaissent. C'est typiquement le type de résultats renvoyés lors d'une requête via un moteur de recherche. Les *highlights* peuvent être vus comme des résumés de documents centrés sur des termes donnés. Le résultat renvoyé est aussi connu sous la dénomination de concordance.

7. Le service de **comparaison de groupes de documents** consiste à représenter les ressemblances et dissemblances entre des collections de documents. Cela se fait en deux étapes. Tout d'abord, nous générons une représentation sous forme d'*embedding* pour chaque groupe (issu d'un groupement choisi par l'utilisateur) en agrégeant les représentations des documents compris dans le groupe. Ces représentations de groupes permettent ensuite de déduire des ressemblances à travers l'algorithme de *clustering* KMeans (Jain, 2010) ou l'analyse en composantes principales (ACP) (Wold et al., 1987). L'analyse par KMeans rassemble les groupes fortement similaires (sémantiquement) dans des *clusters* dont le nombre est défini par l'utilisateur. L'ACP permet quant à elle de visualiser en deux dimensions la similarité entre les différents groupes.

7.4.5 Services d'agrégation et d'analyse de contenus tabulaires

Nous proposons plusieurs façons d'extraire des informations à partir des documents tabulaires dans AUDAL.

8. Le service d'**interrogation SQL** permet aux utilisateurs d'extraire des informations à partir des documents tabulaires, à travers le langage SQL. Les requêtes s'exécutent en réalité sur les représentations raffinées des documents tabulaires et, comme AUDAL intègre ces représentations raffinées dans un SGBD relationnel, il supporte toutes les fonctionnalités d'interrogation SQL, y compris les jointures et les agrégations. Dans l'interface graphique d'AUDAL, les utilisateurs peuvent définir des requêtes SQL de façon assistée et intuitive. Pour ce faire, l'outil leur permet de sélectionner les documents tabulaire à interroger, les opérations de jointure et d'agrégation éventuelles, ainsi que les colonnes à conserver, le tout à l'aide d'un ensemble interactif de listes de choix et de cases à cocher.

9. Le service d'**analyse de corrélations** permet d'évaluer le liens entre une paire de colonnes appartenant à un même document tabulaire. Nous utilisons pour cela une mesure statistique adaptée aux types de colonnes, plus précisément le coefficient de corrélation de Spearman (Costa, 2015) lorsque les deux colonnes sont numériques, et la statistique du Khi-deux (Tallarida & Murray, 1987) lorsque les deux colonnes sont textuelles.

10. Le service de **comparaison de n-uplets** consiste à exécuter un *clustering* KMeans ou une ACP sur un ensemble de n-uplets, en ne prenant en compte que les valeurs numériques. Les n-uplets à comparer sont extraits par une requête SQL, incluant potentiellement des jointures ou des agrégations SQL.

7.5 Discussion

Dans cette section, nous démontrons l'apport et la faisabilité de notre approche d'implémentation en analysant les fonctionnalités d'AUDAL au regard de l'état de l'art.

Il existe dans la littérature plusieurs lacs de données comparables à AUDAL. Ces systèmes peuvent être regroupés en deux catégories : ceux qui se focalisent sur les tâches de recherche de données et ceux qui abordent la problématique d'analyse du contenu des données.

Recherche de données dans les lacs

Une grande partie de la littérature considère les lacs de données comme un bac à sable dédié aux *data scientists*. Ces travaux se concentrent sur la problématique de recherche de données, puisque les analyses portant sur le contenu peuvent être effectuées par des utilisateurs experts. Nous identifions dans la littérature trois approches principales pour prendre en charge la recherche de données dans les lacs de données, à savoir la navigation, la recherche de données connexes et la recherche par mots clés.

En ce qui concerne la **recherche par navigation**, Nargesian et al. (2018a) introduisent un modèle qui exploite un ensemble de *tags* pour partitionner les données contenues dans un lac. Une approche similaire est également mise en œuvre dans GOODS (A. Y. Halevy et al., 2016) et d'autres lacs de données (Bagozi et al., 2019 ; Mehmood et al., 2019). Cependant, toutes ces implémentations considèrent un contexte de données uniquement structurées.

La **recherche de données connexes** exploite des métadonnées sur la similarité entre des entités de données d'un lac ou entre leurs composants (colonnes ou tags). Plusieurs méthodes ont été proposées pour détecter cette similarité dans le contexte des données tabulaires. Maccioni et Torlone (2018) utilisent des dépendances fonctionnelles de type clés primaires-clés étrangères pour associer des documents tabulaires. D'autres approches sont très similaires (Bogatu et al., 2020 ; Farrugia et al., 2016 ; Fernandez et al., 2018). À notre connaissance, seuls Diamantini et al. (2018) abordent la problématique de recherche de données connexes dans un contexte de données non structurées.

La **recherche par mots clés** est le plus souvent associée aux données non structurées, textuelles notamment. Ainsi, nous avons introduit dans des travaux antérieurs le lac de données CODAL, qui utilise un système d'indexation pour optimiser la recherche de données dans les documents textuels (Sawadogo et al., 2019a). Cette méthode a également été mise en œuvre dans un contexte de données structurées par Beheshti et al. (2018) et A. Y. Halevy et al. (2016).

Analyse de contenus dans les lacs de données

A contrario de la vision du lac comme un bac à sable réservé aux *data scientists*, une alternative (à laquelle nous adhérons) propose d'ouvrir le lac de données aux utilisateurs métiers. Suivant cette logique, il est alors nécessaire d'aller au delà de la simple recherche de données pour permettre et faciliter l'analyse du contenu des données.

Une partie des travaux qui abordent cette problématique se concentre sur l'interrogation automatique de données structurées. Malysiak-Mrozek et al. (2018) proposent un système basé sur le langage SQL pour l'interrogation floue des données d'un lac. Dans la même lignée, Hai et al. (2018) introduisent un système de réécriture de requêtes permettant d'interroger de façon uniforme plusieurs systèmes de stockage dans un lac de données. Toujours dans la même logique, Bagozi et al. (2019) proposent une méthode permettant de personnaliser l'interrogation de tables en tenant compte du profil de l'utilisateur. Toutefois, les travaux relatifs à l'analyse du contenu de données semi-structurées et non structurées sont beaucoup moins nombreux. On peut cependant citer les lacs de données Constance (Hai et al., 2016) et CODAL (Sawadogo et al., 2019a).

Bilan

Comme nous l'avons relevé dans la Section 7.5, les propositions de lacs de données se focalisent dans leur grande majorité soit sur la recherche de données, soit sur l'analyse du contenu des données. Chacune de ces propositions aborde ainsi, à notre avis, une vision pertinente, mais partielle des lacs de données. Bien que Bagozi et al. (2019) proposent un système qui s'affranchit de ce clivage, son approche ne prend pas en charge les données non structurées.

D'ailleurs, les données non structurées sont assez rarement prises en charge dans les lacs de données. Nous sommes partis de ce constat pour proposer CODAL pour la gestion des documents textuels dans les lacs de données (Sawadogo et al., 2019a), mais cette première proposition élaborée en Master s'est montrée limitée face à la problématique de passage à l'échelle. De plus, CODAL ne prend en charge que des données non structurées.

AUDAL va au delà de ces limitations, en intégrant aussi bien des services de recherche de données que des services d'analyse de contenus. De plus, il prend en charge à la fois des documents tabulaires *et* des documents textuels, dont l'inclusion dans les lacs de données reste un défi d'actualité.

7.6 Conclusion

Dans ce chapitre, nous avons présenté le lac de données AUDAL. À travers AUDAL, nous avons défini une approche méthodologique pour concevoir et mettre en œuvre un lac de données contenant à la fois des données structurées tabulaires et des données non structurées textuelles. L'approche que nous avons proposée contribue à la littérature à travers une large panoplie d'analyses allant de la recherche de données à l'analyse de contenus textuels et tabulaires.

Ces analyses sont prises en charge grâce à une vision étendue des métadonnées et à une organisation des métadonnées basée sur le modèle MEDAL. Nous distinguons ainsi trois catégories de métadonnées : les métadonnées intra-entité, inter-entités et globales. Les métadonnées intra-entité incluent des représentations raffinées de données brutes, qui permettent d'assister et de faciliter les analyses en transformant systématiquement les données. Les métadonnées inter-entités comprennent des groupements, qui servent à rassembler des documents partageant des caractéristiques communes, ainsi que des relations

de similarité permettant de mesurer les ressemblances entre les documents. Enfin, les métadonnées globales sont des index inversés et des ressources sémantiques qui permettent de filtrer les documents du lac en fonction de la présence ou de l'absence de termes.

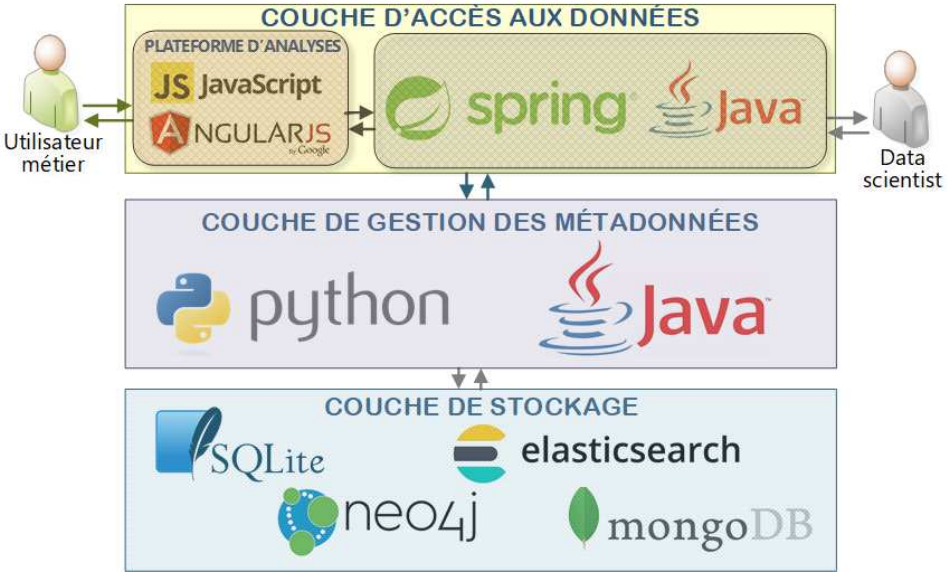


FIGURE 7.12 – Technologies utilisées dans AUDAL

Du point de vue des choix technologiques, AUDAL utilise une combinaison de cinq systèmes de stockage. Un système de fichiers classique (sous Linux) permet de stocker les données brutes. Le SGBD Neo4J sert à matérialiser les relations entre les données et les métadonnées. En d'autres termes, il stocke la structure du système de métadonnées ainsi que les propriétés. Les instances de métadonnées dont le format n'est pas pris en charge par Neo4J sont stockées dans les SGBD MongoDB et SQLite. Enfin, le système d'indexation Elasticsearch permet la constitution d'index inversés. La Figure 7.12 fait un inventaire plus complet des technologies utilisées dans AUDAL.

Nous avons implémenté notre approche de lac de données en utilisant les données du projet AURA-PMI, dans laquelle cette thèse se déroule. Elle vise à traiter à une double problématique. D'une part, nous entendons répondre aux besoins d'analyse, spécifiques au projet AURA-PMI à travers un lac de données adapté. D'autre part, nous visons de façon plus générale à remédier aux limites des approches existantes de modélisation et de mise en œuvre de lacs de données. Nous avons ainsi souligné dans la Section 7.5 le « delta » entre AUDAL et les lacs de données similaires de la littérature.

Chapitre 8

Banc d'essais pour l'évaluation quantitative de lacs de données DLBench

Sommaire

8.1 Introduction

Dans le chapitre précédent, nous avons introduit le lac de données AUDAL et notre approche d'implémentation. Afin d'évaluer les performances d'AUDAL, notamment en termes capacité de de passage à l'échelle, nous avons besoin d'une méthode d'évaluation. Dans un contexte de gestion de données, pour une évaluation quantitative et compte tenu du volume modeste des données utilisées au sein du projet AURA-PMI (6,2 Go), nous avons opté pour un banc d'essais (*benchmark*).

Au delà de la problématique d'évaluation du système AUDAL, nous proposons un banc d'essais qui répond à un besoin de critères d'évaluation partagés dans le contexte des lacs de données. En effet, les lacs de données manquent de standards d'évaluation. Les différentes propositions d'approches d'implémentation sont en effet évaluées de manière hétérogène, à tel point que ces systèmes sont difficilement comparables entre eux. Par conséquent, il est difficile pour les constructeurs de lacs de données d'identifier objectivement les meilleures approches applicables à leurs cas d'usage. Bien qu'il existe des bancs d'essais dans la littérature pour l'évaluation de systèmes *big data* (dont les lacs de données font partie), ces propositions se révèlent cependant limitées et inadaptées par rapport à la panoplie d'analyses possibles dans les lacs de données (Section 8.2).

C'est pourquoi nous proposons le banc d'essais DLBench (*Data Lake Benchmark*), qui se focalise sur les cas d'usage spécifiques aux lacs de données. Dans cette première version, nous nous concentrons plus particulièrement sur les contenus textuels et tabulaires, qui nous intéressent au premier chef. DLBench est *technology-agnostic*, c'est-à-dire qu'il se concentre sur un objectif de gestion des données indépendamment des technologies sous-jacentes (Darmont, 2019).

DLBench comprend :

— un modèle de données,

— un modèle de charge,

— des métriques d'évaluation des performances,

— un protocole de mise en œuvre.

Le modèle de données que nous proposons est composé d'un ensemble documents textuels au format PDF et des documents tabulaires au format CSV. Un paramètre de facteur d'échelle *SF* (*Scale Factor*) permet de faire varier la taille des données dans des proportions prédéterminées, de sorte à analyser la façon dont le système évalué réagit lorsque de le volume de données augmente. Le modèle de charge un ensemble de tâches adaptées au contexte des lacs de données à contenus textuels ou tabulaires. Les métriques d'évaluation permettent de percevoir les bénéfices et les coûts engendrés par le système en termes de stockage et de durée d'exécution des requêtes. Enfin, le protocole d'exécution définit la procédure d'application du banc d'essais.

Un article de conférence internationale décrivant DLBench a été accepté pour publication sous la référence Sawadogo, P. N., Darmont, J. & Noûs, C. (2021a). Benchmarking Data Lakes Featuring Structured and Unstructured Data with DLBench. *Proceedings of the 23rd International Conference on Big Data Analytics and Knowledge Discovery (DaWaK2021), Linz, Austria.*

La suite de ce chapitre est organisée comme suit. Dans la Section 8.2, nous faisons une synthèse des travaux connexes à notre proposition de banc d'essais. Pour ce faire, nous analysons les approches d'évaluation utilisées dans le contexte des lacs de données, d'une part, et celles proposées pour les systèmes *big data* de façon générale, d'autre part. Dans la Section 8.3 nous détaillons les spécifications de DLBench à travers le modèle de données, le modèle de charge, les métriques et le protocole d'évaluation. Enfin, nous discutons la conception de DLBench dans la Section 8.4 en la confrontant à la définition d'un « bon » banc d'essais d'après les critères de Gray (1993).

8.2 État de l'art

Dans cette section, nous balayons les approches communément utilisées pour évaluer les propositions de lacs de données en particulier (Section 8.2.1), et les systèmes d'analyses et de stockage en général (Sections 8.2.2 et 8.2.3).

8.2.1 Approches d'évaluation de lacs de données

Il existe dans la littérature une grande diversité d'approches pour l'évaluation des implémentations de lacs de données. Ces approches se distinguent notamment par la taille des données utilisées, allant de quelques mégaoctets (Alrehamy & Walker, 2015) à plusieurs téraoctets (Maccioni & Torlone, 2018). La différence la plus notable porte cependant sur la forme de l'évaluation. Les différentes approches proposées peuvent ainsi être rassemblées en trois catégories.

Une première approche d'évaluation se base sur l'**expérience utilisateurs**, c'est-à-dire, une appréciation des utilisateurs du système. À ce titre, Fernandez et al. (2018) évaluent leur proposition de lac de données à travers la *plus-value informationnelle* et le *gain de temps* apportés. La plus-value informationnelle représente la quantité de connaissances nouvelles apportées par l'utilisation du système. Le gain de temps désigne quant à lui la différence entre le temps nécessaire pour effectuer une tâche sur le système et le temps nécessaire pour réaliser la même tâche en dehors du système. Cette évaluation se fait à travers un questionnaire rempli par les utilisateurs. Une approche similaire est adoptée par Bagozi et al. (2019), à la différence que ces derniers évaluent plutôt l'*utilisabilité* du système, c'est-à-dire sa facilité de prise en main et son caractère intuitif.

Une deuxième approche d'évaluation consiste à mesurer l'**efficacité de processus et d'algorithmes** qui s'exécutent dans le lac de données. Fernandez et al. (2018) et Bogatu et al. (2020) évaluent ainsi leurs systèmes respectifs en mesurant la *précision* et le *rappel* des processus de détection des liaisons fonctionnelles de type clé primaire-clé étrangère. Mehmood et al. (2019) utilisent quant à eux le critère de l'*aire sous la courbe* (AUC [1]) pour évaluer leur système à travers l'efficacité d'une tâche de classification. Maccioni et Torlone (2018) proposent un critère de mesure d'efficacité composite plus sophistiqué qui fait la synthèse de la précision et du rappel en fonction du temps d'exécution.

1. L'AUC est une métrique de performance communément utilisée lors des tâches de classification. Sa valeur comprise entre 0 et 1 permet de comparer un classifieur au classifieur aléatoire (AUC = 0,5). Plus l'AUC est élevé, plus le classifieur est considéré comme performant.

Une troisième approche d'évaluation consiste à mesurer la **complexité temporelle** de tâches exécutées dans le lac de données. Cette complexité peut être exprimée de façon théorique en déterminant les classes de complexité (logarithmique, linéaire, quadratique, exponentielle, etc.), comme dans les travaux d'Alrehamy et Walker (2015). Elle peut également être mesurée de façon empirique. C'est d'ailleurs cette option qu'ont adopté Nogueira et al. (2018) pour évaluer les performances induites par leur système de métadonnées de lac de données.

8.2.2 Bancs d'essais de systèmes *big data*

Il existe une telle diversité de systèmes et de technologies *big data* qu'aucun critère d'évaluation ne peut raisonnablement les cibler tous (Bajaber et al., 2020). C'est pourquoi il existe des bancs d'essais très divers pour l'évaluation des outils *big data*. Une grande partie de ces bancs d'essais provient du *Transaction Processing Performance Council* (TPC), une organisation spécialisée dans l'évaluation de performances des systèmes décisionnels.

Les bancs d'essais TPC-H (TPC, 2014) et TPC-DS (TPC, 2020a) sont ainsi très utilisés pour évaluer les systèmes décisionnels traditionnels. Ils sont composés chacun d'un modèle de données qui reflète un entrepôt de données et d'une charge de requêtes OLAP typiques, principalement exprimées en SQL. BigBench (Ghazal et al., 2013) est un autre banc d'essais de référence qui traite des requêtes SQL sur les entrepôts de données. Plus récemment, des variantes de BigBench ont inclus des tâches d'analyse plus complexes, notamment l'analyse de sentiments sur des textes courts (Ghazal et al., 2017 ; Ivanov et al., 2020) .

Le banc d'essais TPCx-HS (TPC, 2018) est dédié de façon plus spécifique à l'évaluation des systèmes basés sur les technologies Hadoop ou Spark. Il propose pour cela une unique tâche de tri en guise de charge. HiBench (Huang et al., 2010) est aussi spécifique aux systèmes utilisant Hadoop ou Spark, mais présente un plus grand éventail de tâches dans sa charge. Il mesure en effet les performances du système à travers un ensemble de dix tâches comprenant des agrégations et des jointures SQL, de la classification, du *clustering* et des opération de tri (Ivanov et al., 2015). Dans le même ordre d'idées, le banc d'essais TPCx-AI (TPC, 2020b), qui est toujours en cours de développement, prévoit davantage de tâches relatives aux algorithmes d'intelligence artificielle dans sa charge.

8.2.3 Bancs d'essais de systèmes incluant des données textuelles

Dans cette partie, nous considérons, d'une part, les bancs d'essais de systèmes *big data* avec une partie conséquente de contenus textuels et, d'autre part, les bancs d'essais purement textuels. Le banc d'essais BigDataBench (Wang et al., 2014) est un bon représentant de la première catégorie. Il comprend en effet un jeu de données textuelles constitué de pages Wikipedia [2], ainsi que des charges classiques de recherche d'information telles que des opérations de tri, de filtrage et de comptage de mots clés.

L'un des les bancs d'essais purement textuels les plus récents est TextBenDS (Truica et al., 2021), qui permet d'évaluer les performances des systèmes d'analyse et de traitement

2. https://en.wikipedia.org/

du texte à travers un jeu de données composé de *tweets* et une charge composée de deux tâches. La première tâche, intitulée *Top-K keywords*, consiste en un comptage de mots clés, tandis que la deuxième, *Top-K documents*, consiste à trier des documents sur la base de la présence ou non de mots clés. D'autres bancs d'essais purement textuels se focalisent sur des tâches de traitement du langage naturel, spécifiques à des langues précises. Zhu et al. (2019) et Fialho et al. (2020) proposent ainsi des bancs d'essais pour évaluer les systèmes de reconnaissance de textes en chinois et en portugais, respectivement.

8.2.4 Discussion

La diversité et l'hétérogénéité des approches d'évaluation des lacs de données illustre la nécessité d'un banc d'essais qui permette de comparer les implémentations de lacs de données entre elles. À notre connaissance, aucun des bancs d'essais existants dans la littérature ne répond convenablement à ce besoin. En effet, ni les bancs d'essais de systèmes *big data*, ni les bancs d'essais textuels susmentionnés ne proposent une charge suffisamment étendue pour refléter toutes la panoplie d'analyses possibles dans les lacs de données. Dans le cas des données structurées, la plupart des charges des bancs d'essais ne prennent en compte que des requêtes SQL (TPC-H, TPC-DS, HiBench). Les opérations plus sophistiquées (classification, *clustering*) restent marginales, alors qu'elles constituent des analyses courantes dans les lacs de données. En outre, la tâche de recherche de données connexes est purement absente, alors qu'elle est particulièrement utile dans les lacs de données.

Les charges associées aux bancs d'essais textuels existants sont également insuffisantes. Certes, l'opération de filtrage par mots clés de BigDataBench et les *Top-K documents* de TextBenDS sont pertinents dans un contexte de lac de données. De même, les opérations de comptage de mots clés sont pertinentes pour évaluer l'agrégation de documents (Huang et al., 2010 ; Truica et al., 2021). Cependant, d'autres opérations tout aussi pertinentes, telles que la recherche de documents connexes ou le *clustering* de documents, manquent à l'appel.

Notre proposition de banc d'essais se distingue ainsi à travers une plus étendue, qui comprend à la fois des opérations de recherche de données et d'analyse du contenu des données. Le modèle de données de DLBench diffère également de la plupart des bancs d'essais de systèmes *big data* en fournissant des documents tabulaires bruts. De ce fait, le banc d'essais induit un défi supplémentaire d'intégration de données, puisque celles-ci doivent au préalable être intégrées dans un système adéquat pour être facilement exploitables. De plus, DLBench comprend un ensemble de documents textuels longs qui induit un défi différent de celui causé par les documents courts comme les *tweets* utilisés par Truica et al. (2021) et les articles Wikipedia utilisés par Wang et al. (2014). Enfin, DLBench est *technology-agnostic*, contrairement à certains bancs d'essais de systèmes *big data* comme TPCx-HS et HiBench, Enfin, DLBench est *technology-agnostic*, contrairement à certains bancs d'essais de systèmes *big data* comme TPCx-HS et HiBench, qui se focalisent sur une technologie particulière.

8.3 Spécifications de DLBench

Dans cette section, nous détaillons notre proposition de banc d'essais en quatre étapes. Premièrement, nous décrivons les jeux de données inclus dans DLBench. Deuxièmement, nous faisons l'inventaire des tâches constituant le modèle de charge. Troisièmement, nous introduisons les métriques d'évaluation. Enfin, nous présentons le protocole prévu pour évaluer ou comparer des lacs de données avec DLBench.

8.3.1 Modèle de données

Description des données

DLBench inclut dans son jeu de données des documents textuels, d'une part, et des documents tabulaires, d'autre part. Les documents textuels sont des articles scientifiques (au nombre de 50 000), dont la longueur va de quelques pages à plusieurs dizaines de pages. Ils sont écrits en français et en anglais. Leur volume global est d'environ 62,7 Go. Les documents tabulaires sont quant à eux des fichiers au format CSV, dérivés synthétiquement de plateformes de données ouvertes du gouvernement canadien. Bien que ces données soient souvent considérées comme structurées, elles induisent un défi d'intégration pour leur analyse dans un lac de données. DLBench inclut jusqu'à 5 000 documents tabulaires, dont le volume est d'environ 1,4 Go de données.

Le volume de données est paramétré à travers un facteur d'échelle SF (*Scale Factor*). Cela est particulièrement utile pour analyser l'évolution des performances d'un système lorsque le volume de données augmente. Le facteur d'échelle SF s'échelonne de 1 à 5. La Table 8.1 présente les nombres et tailles des documents inclus dans le jeu de données de DLBench pour chacune des cinq valeurs possibles de SF.

TABLE 8.1 – Volume de données en fonction du facteur d'échelle

Facteur d'échelle	**SF=1**	**SF=2**	**SF=3**	**SF=4**	**SF=5**
Nb. de documents textuels	10 000	20 000	30 000	40 000	50 000
Nb. de documents tabulaires	1 000	2 000	3 000	4 000	5 000
Taille des documents textuels (Go)	8,0	24,9	37,2	49,6	62,7
Taille des documents tabulaires (Go)	0,3	0,6	0,8	1,1	1,4

Le jeu de données inclus dans DLBench est accompagné de deux catalogues de métadonnées qui peuvent faciliter l'intégration des données dans un système de métadonnées. Plus concrètement, nous générons pour les documents textuels un premier catalogue contenant des informations de base à savoir l'année d'édition du document, la langue d'écriture et le domaine de recherche auquel l'article est associé (informatique, sociologie, littérature, etc.). De même, nous proposons pour les documents tabulaires un catalogue qui associe à chaque document un identifiant et une année d'édition. Ces deux catalogues de métadonnées peuvent servir de base aux tâches de recherche de données.

Les deux catalogues de métadonnées restent tout de même indépendants, à l'instar des données auxquelles ils sont associés. Par conséquent, chaque type de données (textuel ou

tabulaire) peut être utilisé indépendamment de l'autre dans le banc d'essais. En d'autres termes, DLBench peut être utilisé pour évaluer un système qui contient des documents textuels uniquement, des documents tabulaires uniquement, ou les deux. On peut ainsi adapter le modèle de données proposé (de même que les tâches inclues dans la charge de travail) à chaque cas d'usage.

Génération des données

La partie textuelle du jeu de données de DLBench est extraite depuis HAL [3], une plateforme française de données ouvertes, dédiée à la diffusion de documents scientifiques. Nous avons fait le choix des documents scientifiques car ils sont pour la plupart suffisamment longs pour engendrer des contraintes analogues à celle attendues dans des cas d'usage réels. Les défis engendrés par le stockage et l'analyse de ces documents contrastent ainsi avec ceux induits par la gestion de documents plus courts comme les *tweets*.

Pour des raisons de droits d'auteur, nous ne sommes pas en mesure de redistribuer les données extraites de HAL, bien que l'accès à HAL soit libre et ouvert. C'est pourquoi, au lieu de distribuer des données extraites de HAL, nous fournissons à la place un script qui permet aux utilisateurs du banc d'essais de le faire eux-mêmes. Ce script et son guide d'utilisation sont disponibles en ligne [4]. Lors de l'extraction des documents textuels, nous nous limitons aux articles scientifiques (nous excluons les mémoires de thèses), dans le but d'obtenir des documents de longueur plus ou moins homogène.

Les données tabulaires sont quant à elles réutilisées à partir d'un banc d'essais pré-existant conçu par Nargesian et al. (2018b). Il s'agit d'un ensemble de 5 000 documents tabulaires synthétiques initialement disponibles sous la forme d'une base de données SQLite. Les tables de cette base de données comportent de nombreuses relations de type clé primaire-clé étrangère. Cela rend ce jeu de données particulièrement approprié pour évaluer l'intégration de données structurées telle qu'elle est effectuée dans les lacs de données. À partir de la base de données SQLite originelle [5], un script permet d'extraire les tables sous forme de fichiers CSV bruts. Comme pour les documents textuels, ce second script et son guide d'utilisation sont disponibles en ligne [6].

8.3.2 Modèle de charge

Pour évaluer et comparer les performances de diverses implémentations de lacs de données, des tâches appropriées et précises sont nécessaires. Dans cette section, nous proposons un ensemble tâches adaptées au contexte des lacs de données, en nous basant sur la littérature. Remarquons que les tâches que nous proposons dans DLBench sont fortement similaires aux services inclus dans l'implémentation d'AUDAL, qui sont aussi inspirés de la littérature.

3. https://hal.archives-ouvertes.fr/
4. https://github.com/Pegdwende44/DLBench
5. https://storage.googleapis.com/table-union-benchmark/large/benchmark.sqlite
6. https://github.com/Pegdwende44/DLBench

Tâches de recherche de données

Ces tâches permettent de trouver des données à partir d'un ensemble de caractéristiques. Trois méthodes sont généralement exploitées pour la recherche de données dans les lacs de données.

1. Le **filtrage par catégories** consiste à sélectionner les documents associés à des valeurs de *tags* ou de propriétés définies dans le catalogue de métadonnées. Cette tâche est souvent présentée comme une opération de navigation à travers les données du lac.
2. La **recherche par mots clés** consiste à retrouver des documents contenant des termes précis. Cette opération est particulièrement pertinente pour les documents textuels, mais peut également servir à retrouver des données tabulaires.
3. La **recherche de données connexes** consiste à récupérer, à partir d'un document spécifié, des documents similaires. Elle peut se baser, par exemple, sur des dépendances fonctionnelles dans un contexte de données tabulaires, ou sur des similarités sémantiques pour des documents textuels.

Tâches d'analyse et d'agrégation de documents textuels

Ces tâches mobilisent le contenu des données. Bien que les documents textuels et les données tabulaires puissent être filtrés avec des méthodes similaires, ils nécessitent des techniques plus spécifiques pour être analysés plus en détails ou agrégés. Ainsi, nous définissons les tâches suivantes pour les documents textuels.

4. Le ***scoring* de documents** est une tâche classique de recherche d'information qui consiste à fournir un score pour chaque document en fonction de sa correspondance avec un ensemble de termes. Il existe une diversité d'algorithmes de calcul de scores, mais tous considèrent entre autres la fréquence d'apparition des termes de requête dans le document et dans le corpus, ainsi que la longueur du document. Cette opération est en réalité équivalente à l'opération *Top-K documents* proposée par Truica et al. (2021).
5. L'**extraction du contexte d'apparition de mots** consiste à extraire de l'ensemble des documents textuels des fragments de textes où certains mots clés apparaissent. C'est aussi une tâche classique de recherche d'information qui fournit une sorte de résumé des documents.
6. L'**extraction des mots les plus fréquents** est une autre méthode communément utilisée pour résumer et agréger des documents (Ravat et al., 2008). Cette opération est donc appropriée pour évaluer des systèmes contenant des documents textuels.
7. La **comparaison de groupes de documents textuels** consiste à appliquer des techniques de fouilles de données pour analyser des contenus textuels agrégés par collections (groupes). Plus concrètement, nous proposons de comparer différents groupes de documents (formés par des *tags*, par exemple) par ACP ou KMeans.

Tâches d'analyse et d'agrégation de documents tabulaires

Ces tâches sont spécifiques à l'analyse de contenus de documents tabulaires.

8. L'**exécution de requêtes structurées simples** permet d'évaluer la capacité d'un système de lac de données à répondre à des requêtes de SGBD, à travers un langage tel que SQL. Comme nous sommes dans un contexte de données tabulaires brutes, le support de ce type d'opérations est un défi en soi.
9. L'**exécution de requêtes structurées complexes** est dans la lignée de la tâche précédente. Elle consiste à évaluer les capacités du système à prendre en charge des requêtes de jointure et de regroupement.
10. La **comparaison de n-uplets** consiste à comparer ou regrouper les n-uplets d'un document tabulaire entre eux. Nous proposons de le faire à travers l'ACP et le *clustering* KMeans. Il s'agit en réalité la même opération que la tâche n° 7, mais les individus statistiques sont ici des n-uplets de documents tabulaires et non des collections de documents textuels.

Nous traduisons les dix tâches qui constituent la charge de DLBench en 20 requêtes concrètes (Table 8.2).

8.3.3 Métriques de performances

Pour évaluer les performances d'une implémentation de lac de données, nous proposons dans cette section trois métriques qui donnent un aperçu de l'efficacité du système en matière de gestion des métadonnées et d'analyses.

1. La **Temps de réponse aux requêtes** vise à mesurer le temps nécessaire pour exécuter individuellement chacune des 20 requêtes que nous avons définies dans la Table 8.2. Cet indicateur (ou plutôt, cet ensemble d'indicateurs) traduit en réalité l'efficacité du système de métadonnées du lac, car c'est ce dernier qui facilite et accélère les analyses. Au cas où certaines requêtes ne soient pas prises en charge, la métrique peut être mesurée en considérant uniquement les requêtes prises en charge.
2. Le **volume des métadonnées** consiste à mesurer l'espace de stockage requis par le système de métadonnées. Cette métrique permet d'observer les performances en termes de temps d'exécution en les confrontant au coût de stockage qui en résulte.
3. La **durée de génération des métadonnées** représente le temps d'exécution du processus de génération de toutes les métadonnées prévues par le système à partir des données brutes. Elle permet également, comme la métrique n° 2, de faire le contre-poids de la métrique n° 1.

8.3.4 Protocole d'évaluation

Les trois métriques que nous avons définies peuvent être mesurées à travers un processus itératif pour chaque facteur d'échelle $SF \in \{1, 2, 3, 4, 5\}$ ou pour un SF donné. Chaque itération se compose de quatre étapes que nous détaillons ci-dessous.

1. **Génération du jeu de données**. Cela consiste à extraire la quantité de données correspondant au facteur d'échelle courant, en se servant des scripts spécifiés dans la Section 8.3.1.
2. **Intégration des données**. Il s'agit d'intégrer les données brutes précédemment générées dans le lac de données à travers la génération et l'organisation d'une couche

TABLE 8.2 – Traduction de la charge de travail de DLBench en requêtes

Tâche	**Requête**	
Recherche de données		
1	Q1A	Retrouver les doc. textuels en *français*
	Q1B	Retrouver les doc. textuels en *anglais*, et édités en *décembre*
	Q1C	Retrouver les doc. textuels associés aux domaines *math* ou *info*, rédigés en *anglais*, et édités en *2010*, *2012* ou *2014*
2	Q2A	Retrouver les documents contenant le terme *university*
	Q2B	Retrouver les documents contenant les termes *university*, *science* ou *research*
3	Q3A	Retrouver les 5 doc. textuels les plus proches d'un document choisi
	Q3B	Retrouver 5 doc. tabulaires qui partagent une relation clé primaire-clé étrangère avec le doc. $t\_dc9442ed0b52d69c\_\_\_\_c11\_1\_\_\_\_1$
Analyses et agrégations de documents textuels		
4	Q4A	Attribuez des scores à tous les doc. textuels en fonction de la présence des termes *university* et *science*
	Q4B	Attribuez des scores à tous les doc. textuels en fonction de la présence des termes *university*, *research*, *new* et *solution*
5	Q5A	Retrouver dans tous les doc. textuels les contextes d'apparition des termes *university* et *science*
	Q5B	Retrouver dans tous les doc. textuels les contextes d'apparition des termes *university*, *science*, *new* et *solution*
6	Q6A	Retrouver les 10 mots clés les plus fréquents dans tous les doc. textuels
7	Q7A	Réaliser une ACP de tous les doc. textuels agrégés par *domaines*
	Q7B	Réaliser un clustering KMeans en 3 classes de tous les doc. textuels agrégés par *domaines*
Analyses et agrégations de documents tabulaires		
8	Q8A	Retrouver tous les n-uplets du doc. tabulaire $t\_e9efd5cda78af711\_\_\_\_c11\_1\_\_\_\_1$
	Q8B	Retrouver les n-uplets de $t\_e9efd5cda78af711\_\_\_\_c11\_1\_\_\_\_1$ dont la colonne *PROVINCE* a la valeur *BC*
9	Q9A	Calculer les valeurs moyennes des colonnes *Unnamed : 12*, *13*, et *20* de $t\_356fc1eaad97f93b\_\_\_\_c15\_1\_\_\_\_1$ en regroupant suivant la colonne *Unnamed : 2*
	Q9B	Réaliser une jointure gauche entre les doc. tabulaires $PED\_SK\_DTL\_SNF\_\_\_\_c7\_0\_\_\_\_1$ et $t\_285b3bcd52ec0c86\_\_\_\_c13\_1\_\_\_\_1$ via leurs colonnes *SOILTYPE*
10	Q10A	Réaliser une ACP sur les résultats de la requête *Q9A*
	Q10B	Réaliser un clustering en 3 classes sur les résultats de la requête *Q9A*

de métadonnées. Cette étape est spécifique à chaque système, car il existe plusieurs approches de gestion des métadonnées.

3. **Mesure du volume et du temps de génération des métadonnées**. Cette étape consiste à calculer l'espace de stockage total utilisé par le système de métadonnées (métrique n° 2), ainsi que la durée d'exécution du processus de génération (métrique n° 3), le tout par rapport au facteur d'échelle courant.

4. **Calcul du temps d'exécution des requêtes.** Il s'agit d'un calcul du temps d'exécution de chacune des 20 requêtes. Pour atténuer les possibles aléas, nous considérons la moyenne de 10 exécutions pour chaque requête. Ces exécutions sont réalisées « à chaud », c'est-à-dire que chaque requête est au préalable exécutée une fois à froid et que son temps de réponse n'est pas pris en compte dans les résultats.

L'algorithme 1 illustre les différentes étapes du protocole d'évaluation.

Algorithm 1: Protocole d'évaluation de DLBench

```
Result: métrique_1, métrique_2, métrique_3
métrique_1 ← [ ][ ] ; métrique_2 ← [ ] ; métrique_3 ← [ ];
for SF ← 1 to 5 do
    génération_des_données(SF);
    génération_et_organisation_metadonnées(SF);
    métrique_2[SF] ← durée_génération_metadonnées(SF);
    metrique_3[SF] ← taille_métadonnées(SF);
    for i ← 1 to 20 do
        exécuter_requête(i, SF);
        durées_exécutions ← [ ];
        for j ← 1 to 10 do
            durées_exécutions[j] ← exécuter_requête(i, SF);
        end
        métrique_1[SF][i] ← moyenne(durées_exécutions);
    end
end
```

8.4 Discussion

Dans cette section, nous confrontons DLBench aux exigences de Gray (1993). Pour ce faire, nous présentons d'abord les critères de conception d'un « bon » banc d'essais de Jim Gray, puis montrons comment DLBench s'y conforme.

8.4.1 Critères de conception d'un bon banc d'essais

D'après Gray (1993), un « bon » banc d'essais doit se conformer aux quatre exigences suivantes.

Pertinence

Un banc d'essais doit cibler des aspects de performance adéquats par rapport au type de système évalué. Ce critère concerne surtout le type de modèle de charge et les métriques du banc d'essais, qui doivent être suffisamment spécifiques pour être adaptés au domaine ciblé.

Portabilité

Un banc d'essais doit fournir un modèle de charge et des métriques suffisamment génériques pour être compatible avec divers (voire à tous les) systèmes relevant du domaine ciblé.

Adaptabilité/Passage à l'échelle

Ce critère concerne surtout le modèle de données du banc d'essais, qui doit s'adapter aux capacités du système évalué. Cela évite que le banc d'essais ne soit inapplicable à des systèmes aux capacités réduites, ou à l'inverse insignifiant pour des systèmes à très grande capacité.

Simplicité

Un banc d'essais doit être simple à comprendre et à mettre en œuvre, sans quoi il risque d'être peu utilisé et de perdre ainsi en crédibilité.

8.4.2 Conformité de DLBench aux critères de Jim Gray

Dans cette section, nous explicitons comment DLBench se conforme aux quatre exigences énoncées dans la Section 8.4.1.

Pertinence

Le modèle de données de DLBench est constitué d'un ensemble de documents tabulaires et textuels que nous jugeons adéquats au contexte des lacs de données. À travers les documents tabulaires nous représentons la majorité des données structurées actuellement inclues dans les lacs de données (Bogatu et al., 2020 ; Fernandez et al., 2018 ; A. Halevy et al., 2016 ; Nargesian et al., 2018a). Les documents textuels reflètent quant à eux la grande majorité des données potentiellement utilisables dans les lacs de données, puisqu'elles constituent la majorité des *big data*. À l'image du modèle de données, le modèle de charge et les métriques que nous proposons sont également spécifiques et ajustés au contexte des lacs de données.

Portabilité

Le modèle de données de DLBench peut être généré à l'aide de simple scripts Python, qui sont fournis. Python étant multi-platefome et libre, les données de DLBench peuvent donc être facilement intégrées dans n'importe quel système, sans limitation. De plus, le modèle de charge a été défini sous la forme de tâches à réaliser, sans contrainte sur les technologies utilisées. Elle est ainsi applicable à divers systèmes et approches d'analyses.

Adaptabilité/Passage à l'échelle

DLBench est paramétré par le facteur d'échelle SF analogue à celui des bancs d'essais du TPC, qui permet de choisir cinq volumes de données croissants pour l'expérimentation. Cela permet d'ajuster le jeu de données aux capacités du système à évaluer. Par ailleurs, le modèle de données que nous proposons peut être remplacé par d'autres jeux de données, réels ou synthétiques, tout en conservant le modèle de charge (les tâches) et en adaptant *a minima* les requêtes correspondantes. Les métriques que nous avons définies sont également génériques.

Simplicité

Le faible nombre de paramètres (seulement 1) et de métriques (seulement 3), ainsi que la structuration du modèle de charge conçu en tâches et requêtes, permet de faciliter la compréhension de DLBench par les utilisateurs. À cette facilité de compréhension s'accompagne une facilité d'implémentation, car les métriques proposées sont faciles à mesurer et les requêtes peu contraignantes en termes de technologie à employer pour les mettre en œuvre.

8.5 Conclusion

Dans ce chapitre, nous avons proposé DLBench, un banc d'essais pour l'évaluation de lacs de données. DLBench comprend un modèle de données, un modèle de charge, un ensemble de métriques et un protocole d'évaluation. Il s'agit à notre connaissance du premier banc d'essais dédié aux lacs de données.

Plus concrètement, le modèle de données de DLBench est constitué d'un ensemble de documents textuels et tabulaires. Les documents textuels proviennent de HAL, la plateforme française de données ouvertes qui rassemble des document scientifiques. Ces documents peuvent être obtenus par l'utilisateur à l'aide de scripts Python que nous fournissons, selon un facteur d'échelle à définir. Les documents tabulaires sont quant à eux des fichiers CSV issus d'un banc d'essais pré-existant proposé par Nargesian et al. (2018b).

Le modèle de charge de DLBench est constitué d'un ensemble de dix tâches que nous avons déclinées en vingt requêtes opérationnelles. Ces tâches sont structurées en trois catégories qui reflètent la panoplie d'analyses possibles dans un lac de données tabulaires et textuelles. Les tâches de recherche de données permettent de retrouver un document à partir d'un ensemble de critères (appartenance à un groupe, présence ou absence d'un terme, ressemblance avec un autre document, etc.). Les tâches d'analyse et d'agrégation de contenus textuels consistent à obtenir des informations sur le contenu d'un ensemble de documents ou sur les interactions entre documents. Enfin, les tâches d'analyse et d'agrégation de contenus tabulaires consistent quant à elles en des requêtes structurées (en SQL, par exemple) ou en des analyses exploitant la fouille de données.

Pour finir, notre proposition de banc d'essais inclut trois métriques et un protocole d'évaluation qui permettent d'analyser les capacités de passage à l'échelle d'un système en l'évaluant avec plusieurs valeurs du facteur d'échelle SF. Ils peuvent également servir à comparer de façon objective des implémentations différentes de lacs de données.

L'évaluation de DLBench montre qu'il se conforme aux caractéristiques d'un « bon » banc d'essais, selon les critères de Gray (1993). Le modèle de données de DLBench peut cependant paraître limité en termes de volume, surtout concernant la partie tabulaire du jeu de données qui est limitée à 5000 documents (1,4 Go). C'est pourquoi nous envisageons dans une prochaine version de substituer la partie tabulaire avec des documents plus nombreux et plus volumineux extraits de plateformes de données ouvertes.

Chapitre 9

Évaluation globale du lac de données AUDAL

Sommaire

9.1 Introduction

Dans le Chapitre 7.1, nous avons présenté AUDAL, notre approche de conception et de mise en œuvre de lac de données. Dans ce chapitre, nous proposons une première évaluation quantitative d'AUDAL (en termes de temps de réponse) en nous servant du jeu de données du projet AURA-PMI. Cependant, le volume limité des données (seulement 6,2 Go) ne permet pas d'évaluer les capacités de passage à l'échelle du système AUDAL. Nous approfondissons donc notre évaluation quantitative en appliquant le banc d'essais DLBench (Chapitre 8.1) à AUDAL. Nous évaluons ainsi de façon plus complète les performances induites par notre approche d'implémentation de lac de données. Nous analysons surtout comment AUDAL passe à l'échelle à la fois en termes de temps de réponse aux requêtes et de gestion des métadonnées, à travers la durée de génération et l'espace de stockage utilisé.

Si cette évaluation quantitative indique les capacités de passage à l'échelle du système, elle ne reflète cependant pas l'appréciation des utilisateurs du système, qui est pourtant également importante. C'est pourquoi, nous proposons une évaluation de l'expérience utilisateur du système AUDAL, notamment en mesurant l'utilisabilité de l'interface graphique, ainsi que la plus-value informationnelle et le gain de temps engendrés par l'utilisation d'AUDAL pour les utilisateurs en regard aux méthodes traditionnellement utilisées. Nous réalisons cette évaluation en adoptant et en adaptant des protocoles basés sur des questionnaires déjà utilisés dans le contexte des lacs de données (Bagozi et al., 2019 ; Fernandez et al., 2018).

Enfin, nous allons au delà du contexte des lacs de données en analysant comment AUDAL se conforme aux caractéristiques attendues de systèmes d'analyse et de visualisation de données. D'une part, nous confrontons AUDAL aux règles de Codd et al. (1993) applicables aux systèmes OLAP, car les analyses textuelles dans AUDAL sont comparables aux analyses OLAP. D'autre part, nous confrontons notre implémentation de lac de données aux caractéristiques édictées par Faber et al. (2019) pour les systèmes modernes de visualisation de données.

9.2 Évaluation quantitative préliminaire

Nous proposons une première évaluation quantitative d'AUDAL en nous servant du jeu de données AURA-PMI. Pour rappel, ce jeu de données est constitué de 8122 documents textuels et de 6 documents tabulaires, le tout représentant une taille de 6,2 Go. Pour les besoins de cette évaluation, nous utilisons un ensemble de 15 requêtes qui reflètent les principales fonctions proposées par AUDAL (Tableau 9.1).

La Table 9.2 présente les temps de réponses des 15 requêtes sous la forme d'une moyenne de 10 exécutions. Ces résultats montrent qu'AUDAL assure une réponse aux requêtes en un temps globalement satisfaisant. Cela est particulièrement vrai pour les requêtes de recherche de données ainsi que les analyses de contenus tabulaires (0,11 s et 0,24 s en moyenne, respectivement).

Cependant, la moitié des requêtes d'analyse de contenus textuels requiert un temps plus conséquent pour s'exécuter (5, 2 et 2 secondes pour les requêtes n° 9, n° 10 et n° 11, respec-

TABLE 9.1 – Requêtes d'évaluation préliminaire d'AUDAL

Requêtes de recherche de données	
1	Filtrer les documents textuels en anglais et édités en décembre
2	Retrouver les documents contenant les termes "big" et "data"
3	Retrouver les documents contenant "big", "data", "document" et "article"
4	Pour un document tabulaire choisi, retrouver 3 autres documents qui peuvent lui être associés (par jointure)
5	Pour un document textuel choisi, retrouver les 5 documents textuels qui lui sont le plus similaires
Requêtes d'agrégation et d'analyse de contenus textuels	
6	Calculer les scores de tous les documents textuels par rapport aux termes "big", "data", "article" et "document"
7	Extraire une concordance des documents par rapport aux termes "data" et "ai"
8	Extraire une concordance des documents par rapport aux termes "data", "ai", "article" et "paper"
9	Afficher les 10 mots les plus fréquents dans l'ensemble des documents textuels
10	Appliquer un KMeans en 3 classes sur les documents regroupés par mois
11	Appliquer une ACP sur les documents regroupés par mois.
Requêtes d'agrégation et d'analyse de contenus tabulaires	
12	Réaliser une jointure entre deux tables
13	Réaliser une jointure entre deux tables en agrégeant toute les valeurs numériques suivant une colonne catégorielle
14	Appliquer un KMeans en 3 classes sur le résultat de la *requête n° 13*
15	Appliquer une ACP sur le résultat de la *requête n° 13*.

tivement), mais cela ne représente qu'une minorité des fonctions du système AUDAL. De plus, ces performances peuvent sans doute être améliorées en augmentant les ressources CPU de l'infrastructure physique qui supporte le système. Enfin, nous notons que sans notre approche, de telles analyses seraient littéralement impossibles pour les utilisateurs métiers.

Ces performances prometteuses sont obtenues au prix d'un système de métadonnées étendu. La Table 9.3 montre en effet que la taille des métadonnées dans AUDAL représente presque la moitié des données brutes (2,8 Go de métadonnées pour 6,2 Go de données brutes). Nous jugeons cela tout de même acceptable, compte tenu des avantages procurés (large panoplie d'analyses, rapidité des analyses et analyses réalisées par des utilisateurs métiers). Il est d'ailleurs admis, en raison de leur importance, que les métadonnées peuvent être très volumineuses dans le contexte des lacs de données (HELLERSTEIN et al., 2017).

TABLE 9.2 – Temps de réponses aux requêtes

Requête	Temps de réponse (ms)
Recherche de données	
Requête #1	194
Requête #2	108
Requête #3	143
Requête #4	59
Requête #5	51
Analyse de contenus textuels	
Requête #6	85
Requête #7	169
Requête #8	62
Requête #9	4 629
Requête #10	1 930
Requête #11	1 961
Analyse de contenus tabulaires	
Requête #12	71
Requête #13	61
Requête #14	174
Requête #15	670

TABLE 9.3 – Données et métadonnées

Système	Taille (Go)
Données brutes	
-	**6,2**
Métadonnées	
Neo4J	0,9
SQLite	0,003
MongoDB	0,28
ElasticSearch	1,6
Total	**2,8**

9.3 Évaluation quantitative avancée avec DLBench

Une première évaluation quantitative de notre implémentation sur la base de requêtes-types montre des résultats prometteurs en termes de performances. Mais cela ne suffit pas à prouver la capacité de passage à l'échelle de notre solution. C'est pourquoi nous proposons dans cette section une évaluation plus approfondie de notre implémentation à l'aide de données artificielles plus volumineuses.

Nous évaluons notre implémentation de lac de données en lui appliquant le protocole d'évaluation de DLBench (Section 8.3.4). Nous présentons et analysons les résultats obtenus en fonction des trois métriques de DLBench.

Pour rappel, AUDAL est basé sur une infrastructure physique constituée de trois machines virtuelles. Une de ces machines, la principale, est dotée de 24 Go de mémoire et d'un processeur à 7 cœurs, tandis que les deux autres ont chacune 24 Go de RAM et un processeur à un seul cœur (Section 7.3.2). Chacune des trois machines héberge une instance d'Elasticsearch, une instance de Neo4j et une instance de MongoDB. La machine principale héberge en plus une instance de SQLite, ainsi que l'application web qui supporte l'API REST et l'interface graphique d'AUDAL. C'est aussi sur cette machine que s'exécute le processus de génération des métadonnées.

9.3.1 Métrique n° 1 : Temps de réponses des requêtes

Nous analysons les temps de réponse obtenus sur les dix tâches de DLBench et déclinées en 20 requêtes (Section 8.3.2).

Tâche n° 1 : Filtrage par catégories

La Figure 9.1 montre l'évolution du temps de réponse des requêtes de filtrage en fonction du facteur d'échelle SF. On constate une évolution linéaire sur les requêtes avec un ou deux axes de filtrage (requêtes Q1A et Q1B). Cette évolution devient cependant quadratique quand la requête est plus complexe (requête Q1C). Les temps de réponse restent tout de même acceptables avec 6.2 secondes de temps de réponse pour la requête la plus complexe sur le facteur d'échelle le plus élevé. Dans le contexte du projet AURA-PMI, les utilisateurs métiers peuvent tolérer en effet quelques secondes de temps de réponse, mais sans doute pas une minute, sauf exceptions.

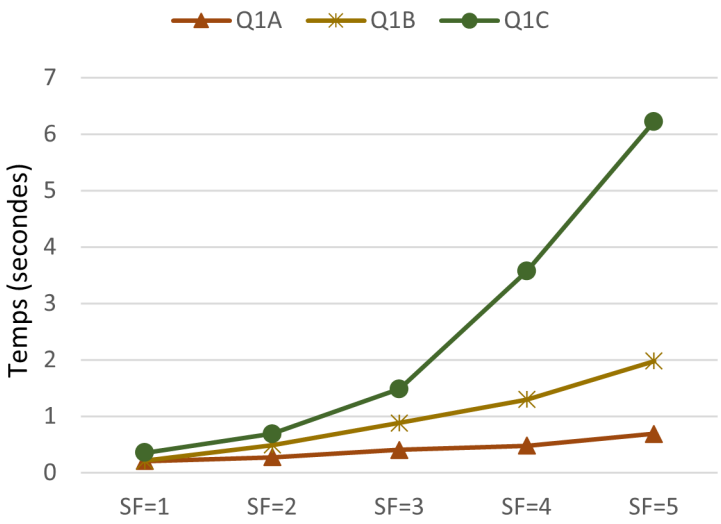


FIGURE 9.1 – Temps d'exécution des requêtes de filtrage par catégories

Tâche n° 2 : Recherche par mots clés

La Figure 9.2 montre l'évolution du temps de réponse des requêtes de recherche par mots clés. On constate une tendance générale linéaire à faible évolution pour la recherche uni-terme (requête Q2A), et une tendance générale linéaire avec une évolution plus importante pour la recherche multi-termes (requête Q2B). L'évolution des temps de réponse est par endroits bruitée (cela se voit à travers une courbe irrégulière). Cela peut être expliqué par l'influence de facteurs extérieurs difficilement contrôlables que sont le trafic sur le réseau hébergeant les machines virtuelles et l'optimisation automatique de mémoire (*garbage collection*) de Java (Section 9.3.4). Cette influence est cependant très faible (de l'ordre d'un dixième de seconde) et ne se voit donc que sur des résultats dont les temps de réponse sont très courts, comme c'est le cas ici.

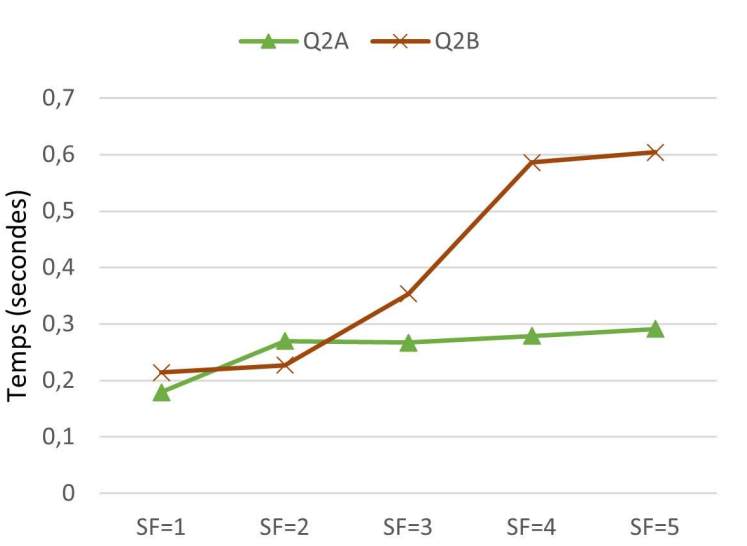


FIGURE 9.2 – Temps d'exécution des requêtes de recherche par mots clés

Tâche n° 3 : Recherche de documents connexes

La Figure 9.3 montre l'évolution du temps de réponse des requêtes de recherche de données connexes. En plus d'être très faibles, les temps de réponse sont constants, à la fois pour les documents textuels (requête Q3A) et pour les documents tabulaires (requête Q3B). Les variations perceptibles peuvent être assimilées à du bruit que nous expliquons dans la Section 9.3.4.

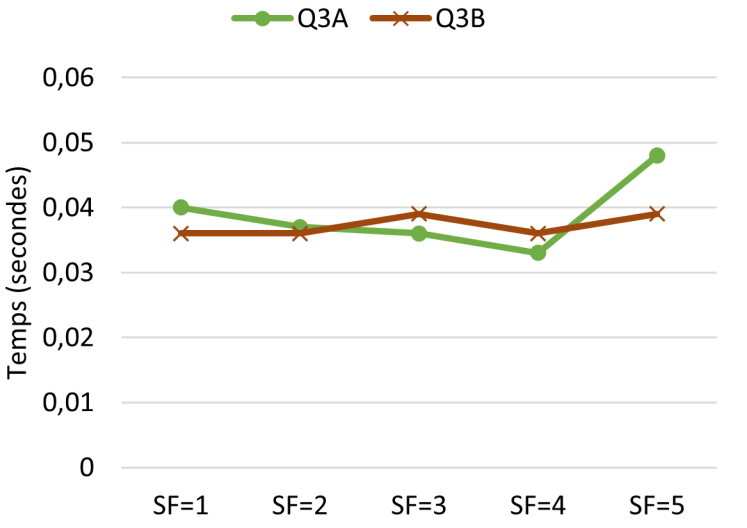


FIGURE 9.3 – Temps d'exécution des requêtes de recherche de données connexes

Tâche n° 4 : Scoring de documents

La Figure 9.4 montre l'évolution du temps de réponse des requêtes de *scoring* de documents textuels. Ces requêtes ont une tendance linéaire, voire logarithmique. Là encore, les temps de réponse sont très faibles (tous inférieurs à 0,16 secondes), ce qui occasionne l'apparition de perturbations.

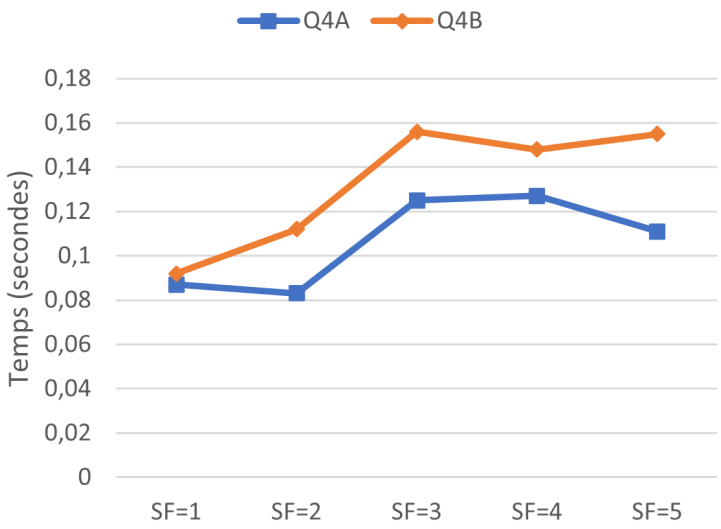


FIGURE 9.4 – Temps d'exécution des requêtes de recherche de données connexes

Tâche n° 5 : Extraction de *highlights*

La Figure 9.5 montre l'évolution du temps de réponse des requêtes d'extraction de *highlights* dans des documents textuels. les temps de réponse restent très faibles (moins de 0,1 seconde) et plus ou moins constants quand le facteur d'échelle augmente. On constate aussi qu'il n'y a pas de grande différence dans les temps de réponse en fonction du nombre de termes de recherche (requêtes Q5A et Q5B, respectivement).

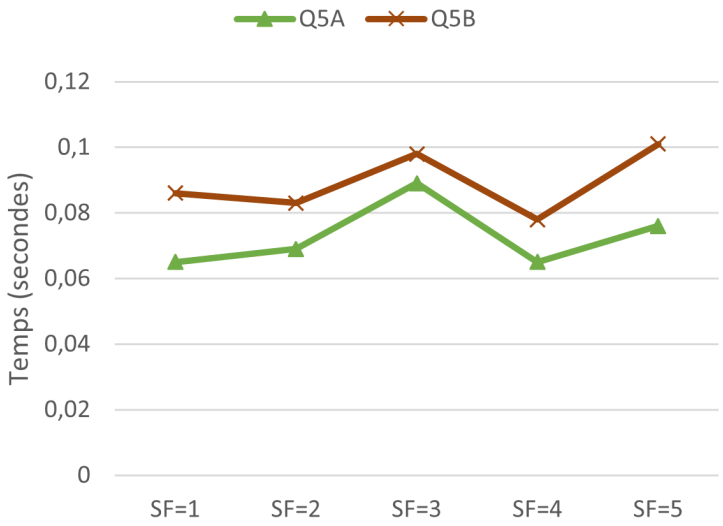


FIGURE 9.5 – Temps d'exécution des requêtes d'extraction de *highlights*

Tâche n° 6 : Recherche des mots clés les plus fréquents

La Figure 9.6 montre l'évolution du temps de réponse des requêtes de comptage des mots clés les plus fréquents. On y perçoit une croissance linéaire des temps de réponse, qui s'échelonnent de 18 secondes pour $SF = 1$ à 104 secondes pour $SF = 5$. Ces temps de réponses sont beaucoup plus élevés que dans les tâches précédentes, mais cela reste tout de même raisonnable à notre avis car ce fait peut être présenté comme un cas particulier.

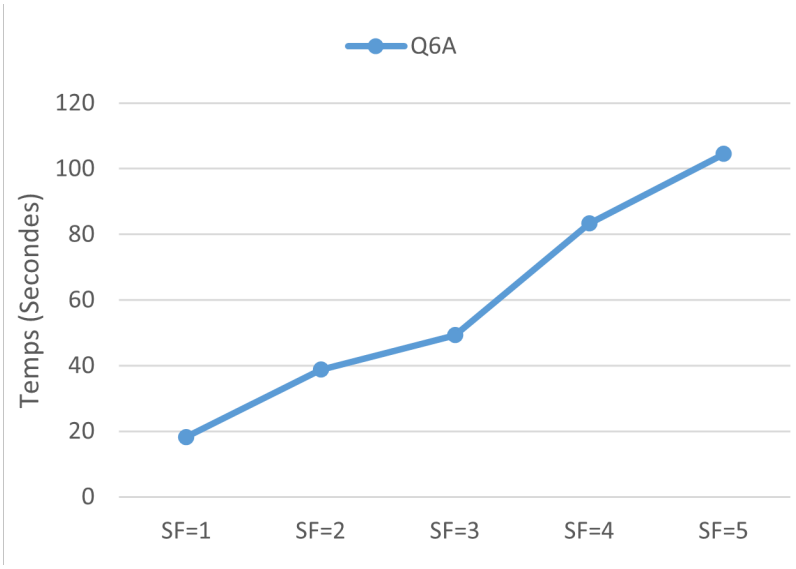


FIGURE 9.6 – Temps d'exécution des requêtes de recherche des mots clés les plus fréquents

Tâche n° 7 : Comparaison de groupes de documents textuels

La Figure 9.7 montre l'évolution du temps de réponse des requêtes de comparaison de groupes de documents textuels. On constate que les temps de réponse évoluent linéairement en fonction du facteur d'échelle. De plus, les deux types de *clustering* (ACP et KMeans) ont des temps d'exécution pratiquement identiques, s'échelonnant de 4 secondes pour $SF = 1$ à 21 secondes pour $SF = 5$. Ces temps de réponses restent de l'ordre de la seconde, et sont donc acceptables pour les utilisateurs.

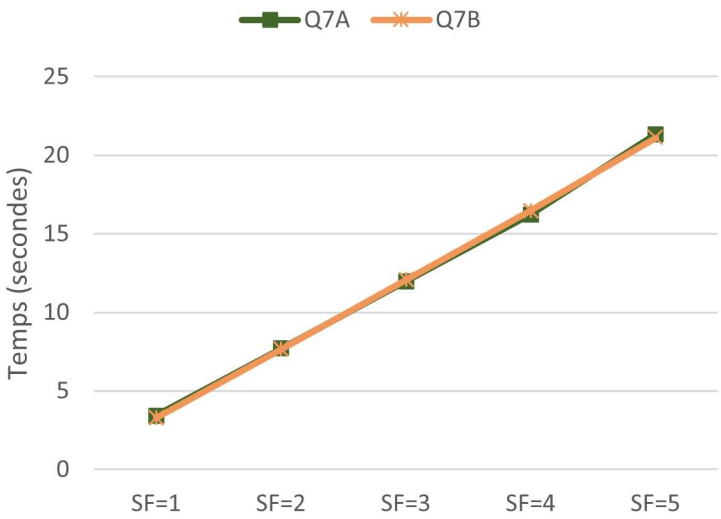


FIGURE 9.7 – Temps d'exécution des requêtes de comparaison de groupes de documents textuels

Tâche n° 8 : Requêtes SQL simples

La Figure 9.8 montre l'évolution du temps de réponse des requêtes structurées simples. Il s'agit dans notre cas de requêtes SQL. On constate des temps de réponse qui ne semblent pas évoluer en fonction du facteur d'échelle. L'influence semble venir principalement du bruit engendré par les facteurs extérieurs (Section 9.3.4).

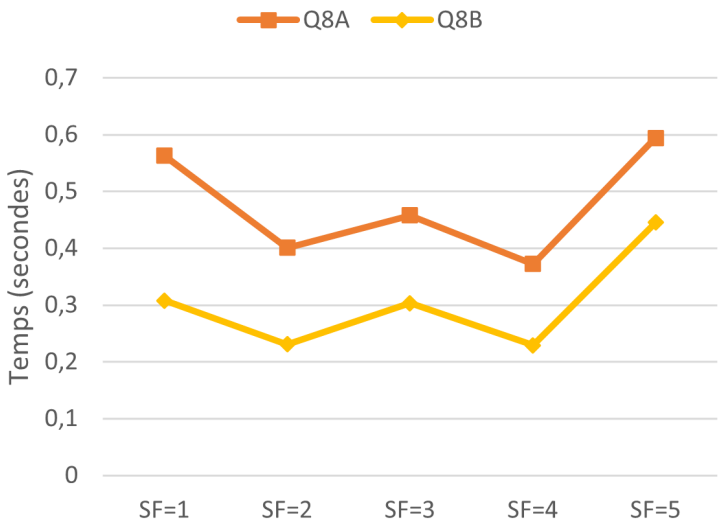


FIGURE 9.8 – Temps d'exécution des requêtes SQL simples

Tâche n° 9 : Requêtes SQL complexes

La Figure 9.9 montre l'évolution du temps de réponse des requêtes structurées complexes. Il s'agit de requêtes SQL de jointure et d'agrégation. Les résultats montrent des temps de réponse très faibles (moins de 0,3 seconde), étonnamment plus faibles (certainement à cause d'un plus faible nombre de n-uplets traités) que dans le cas des requêtes simples. Les seules variations semblent liées aux influences externes.

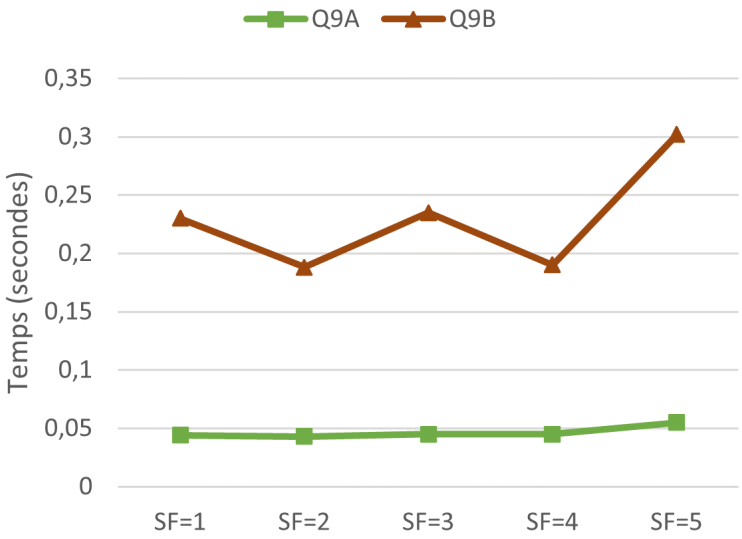


FIGURE 9.9 – Temps d'exécution des requêtes SQL complexes

Tâche n° 10 : Requêtes de comparaison de n-uplets

La Figure 9.10 montre l'évolution du temps de réponse des requêtes de comparaison de n-uplets de documents tabulaires. Là encore, les temps de réponse sont faibles et demeurent constants avec l'évolution du facteur d'échelle. On note tout de même que l'exécution de l'ACP (requête Q10A, 0,14 seconde en moyenne) est largement plus rapide que celle du KMeans (requête Q10B, 0,46 seconde en moyenne). Cela est probablement imputable à la différence de complexité entre les deux algorithmes (ACP et KMeans).

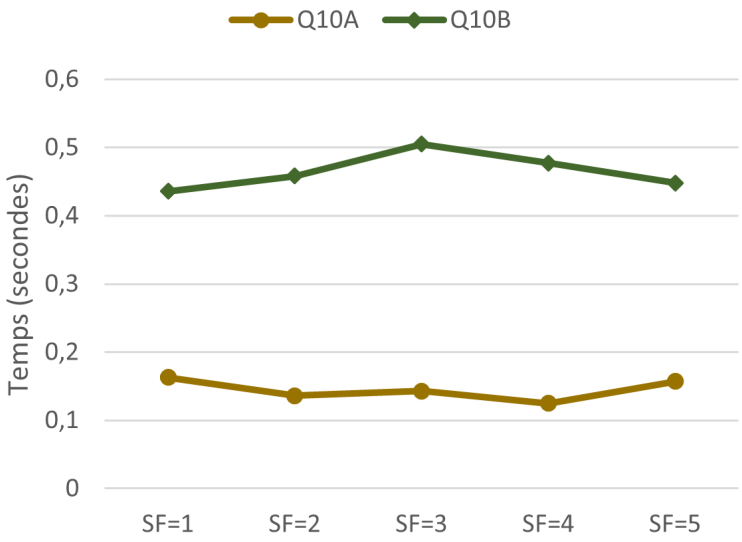


FIGURE 9.10 – Temps d'exécution des requêtes de comparaison de n-uplets de documents tabulaires

9.3.2 Métrique n° 2 : Taille des métadonnées

La taille des métadonnées évolue de façon logarithmique avec le facteur d'échelle, contrairement à la taille des données brutes, qui évolue linéairement. En comparant la taille des métadonnées à celle des données brutes, nous nous apercevons en effet que le ratio métadonnées/données décroît avec le facteur d'échelle (Figure 9.11). Pour preuve, la taille des métadonnées va de 83 % pour $SF = 1$ à 54 % des données brutes pour $SF = 5$.

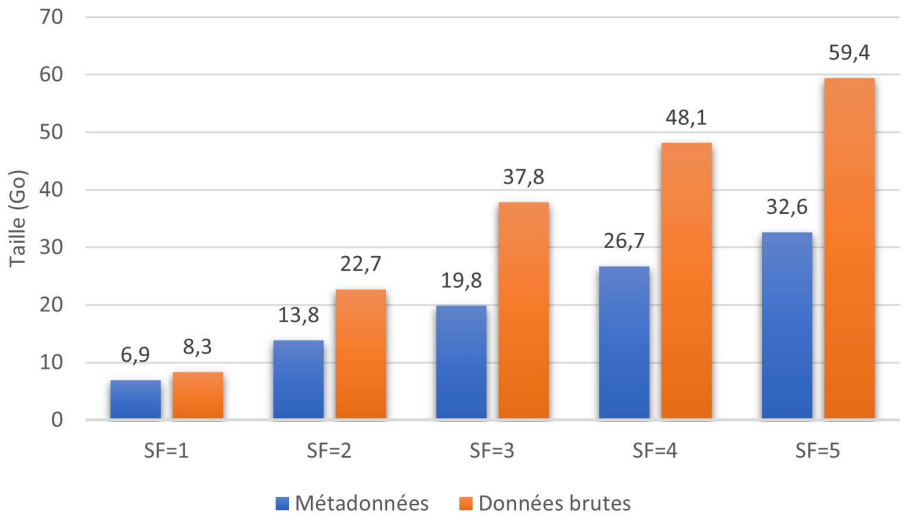
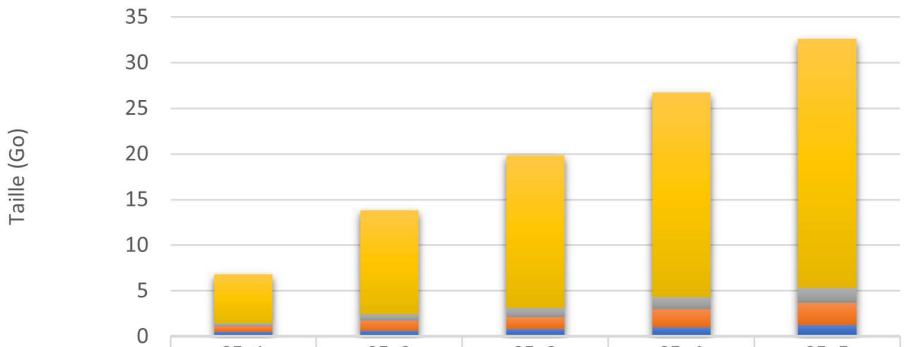


FIGURE 9.11 – Comparaison de la taille des métadonnées à celle des données brutes

En analysant la distribution des métadonnées à travers les quatre systèmes de stockage utilisés dans AUDAL (Elasticsearch, SQLite, MongoDB et Neo4j – Figure 9.12), nous constatons que l'écrasante majorité des métadonnées (en termes de taille) est représentée par le stockage sous Elasticsearch (27,3 Go, soit 83 % des métadonnées pour $SF = 5$). On peut en déduire que notre stratégie d'indexation coûte relativement cher.



	SF=1	SF=2	SF=3	SF=4	SF=5
Elasticsearch	5,4	11,4	16,7	22,4	27,3
SQLite	0,3	0,7	1	1,3	1,6
MongoDB	0,6	1,1	1,3	2	2,5
Neo4J	0,5	0,6	0,8	1	1,2

FIGURE 9.12 – Distribution des métadonnées à travers les systèmes de stockage

9.3.3 Métrique n° 3 : Durée de génération des métadonnées

Comme l'illustre la Figure 9.13, le temps de génération des métadonnées évolue de façon exponentielle. Il s'échelonne ainsi de 3,4 heures pour $SF = 1$ à 72 heures pour $SF = 5$.

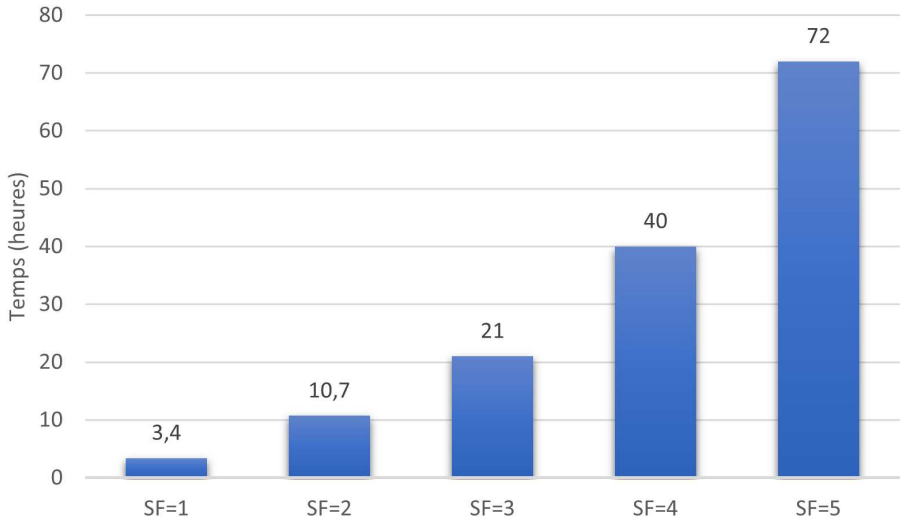
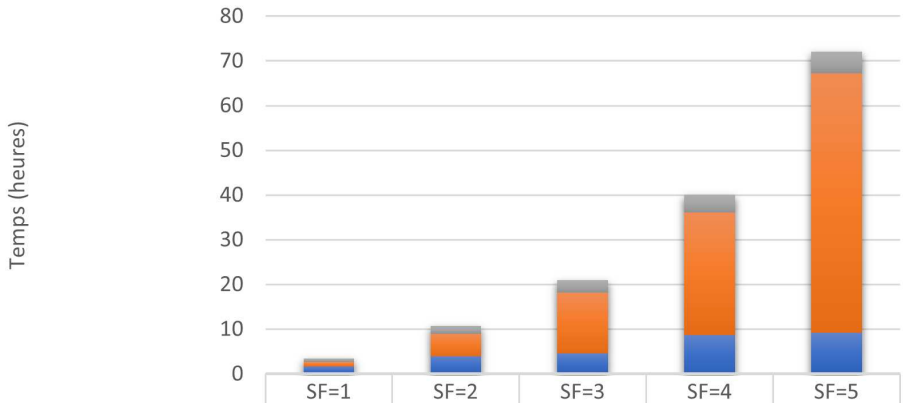


FIGURE 9.13 – Temps de génération des métadonnées

En analysant la distribution des temps de génération par type de métadonnées (Figure 9.14), nous constatons que la croissance exponentielle est principalement du fait des métadonnées inter-entités. On remarque ainsi que le temps de génération et la proportion entre les métadonnées inter-entités et toutes les métadonnées s'échelonnent de 27 % pour $SF = 1$ à 80 % pour $SF = 5$.



	SF=1	SF=2	SF=3	SF=4	SF=5
Métadonnées globales	0,7	1,8	2,9	3,9	4,8
Métadonnées inter	1,1	5	13,5	27,4	58
Métadonnées intra	1,6	3,9	4,6	8,7	9,2

FIGURE 9.14 – Distribution du temps de génération par type de métadonnées

9.3.4 Présence de bruit dans les temps de réponse

Lors d'une expérience préalable, nous avons constaté une forte variation du temps de réponse à la charge de DLBench en fonction de l'heure de la journée. Cela nous a convaincu que le trafic sur le réseau de l'Université Lyon 2 influence les temps de réponse, puisque les machines virtuelles qui supportent le système AUDAL sont hébergées à la Direction des

systèmes d'information de l'université, au même titre que d'autres applications destinées aux étudiants et personnels. Pour limiter l'influence de ce facteur, nous avons choisi une plage horaire uniforme et tardive pour l'exécution des requêtes : entre 22h00 et 23h59. Nous avons ainsi réduit les perturbations externes et la présence de bruit dans les temps d'exécution de nos requêtes.

En plus du trafic sur le réseau, d'autres perturbations pourraient s'expliquer par l'exécution de l'optimisation automatique de la mémoire (*garbage collection*) par Java. Ce processus, dont le déclenchement est plus ou moins aléatoire, influence les performances du système selon qu'il arrive avant, pendant ou après une tâche. L'influence de ces deux facteurs est toutefois minime, puisqu'elle est seulement de l'ordre du dixième de secondes. Pour preuve empirique, on ne remarque pas de bruit sur les requêtes dont le temps d'exécution excède une seconde.

9.3.5 Discussion

Les résultats de cette évaluation quantitative montrent qu'AUDAL passe globalement bien à l'échelle. Les temps de réponse aux requêtes sont pour la plupart stables (tâches n° 3, n° 5, n° 8, n° 9 et n° 10) ou évoluent de façon linéaire ou logarithmique (tâches n° 2, n° 4, n° 6 et n° 7) Seule une requête de la tâche n° 1 suit une tendance exponentielle. Cette dernière conserve tout de même un temps d'exécution raisonnable (6,2 secondes) pour $SF = 5$.

La requête Q6A de la tâche n° 6 a le temps d'exécution le plus important (104 secondes pour $SF = 5$). En plus de cette tâche, seules les tâches n° 1 et n° 7 ont des temps de réponse qui excèdent 1 seconde. Les temps de réponse sont donc pour la plupart faibles. Même les temps de réponse des tâches n° 1 et n° 7, qui sont les plus élevés, restent raisonnables au vu des caractéristiques de l'infrastructure de stockage utilisée et du caractère exceptionnel (Table 7.1). De plus, ces tâches n'ont pas vocation à être utilisées sur l'ensemble du jeu de données mais plutôt sur un sous-ensemble, contrairement aux tâches de recherche de données. Nous notons enfin que, sans notre interface graphique, ces tâches seraient simplement impossibles à effectuer par les utilisateurs métiers du projet AURA-PMI.

Ces performances en temps de réponse, que nous estimons prometteuses, sont toutefois obtenues au prix d'un espace de stockage élevé (pour les métadonnées) et d'un long processus de génération des métadonnées. Le coût en termes de volume de stockage nous paraît acceptable au vue des bénéfices engendrés, et surtout au vue de son évolution avec le facteur d'échelle, qui paraît soutenable. Tel n'est pas le cas du processus de génération des métadonnées qui, lui, évolue de façon exponentielle. Il nous paraît donc indispensable d'optimiser la génération des métadonnées inter-entités, qui est la source de cette explosion combinatoire.

9.4 Évaluation de l'expérience utilisateur

En complément de l'évaluation quantitative de performances menée à la Section 9.3, nous proposons dans cette section une évaluation basée sur l'expérience utilisateur. Il s'agit plus

concrètement de recueillir la perception des utilisateurs sur la qualité de l'outil AUDAL. Pour ce faire, nous adoptons et adaptons des approches utilisées dans la littérature sur les lacs de données. Nous évaluons ainsi l'utilisabilité de l'outil, la plus-value informationnelle engendrée et le gain de temps apporté.

9.4.1 Méthodologies d'évaluation de l'expérience utilisateur

Notion d'utilisabilité

La notion d'utilisabilité est assez abstraite. Par conséquent, elle est difficile à évaluer et plus encore à mesurer. Il existe tout de même dans la littérature scientifique et surtout industrielle des approches standards, communément utilisées et acceptées pour répondre à ce besoin.

Nous adoptons pour notre part l'approche SUS (*System Usability Scale*), qui est l'une des méthodes les plus citées dans la littérature et utilisée en pratique. Elle a en effet été employée comme méthodologie de post-évaluation dans 43 % des projets informatiques industriels en 2009 (Lewis, 2018). De plus, c'est à notre connaissance la seule approche utilisée pour évaluer l'utilisabilité de lacs de données (Bagozi et al., 2019).

La méthodologie d'évaluation SUS a été introduite par Brooke (1996). Elle consiste à proposer un questionnaire aux utilisateurs, pour ensuite agréger les réponses en un score d'utilisabilité. Plus ce score est élevé, plus le système est considéré comme utilisable, et inversement. Le questionnaire typique associé à la méthode d'évaluation SUS comprend dix affirmations qui ciblent de façon alternée les points positifs et négatifs du système évalué. L'utilisateur répond à chacune des affirmations en attribuant une valeur (appréciation) comprise entre 1 et 5, selon qu'il est en accord (5 pour un accord total) ou en désaccord (1 pour un désaccord total). L'ensemble des dix réponses permet d'obtenir un score compris entre 0 et 100, en utilisant les formule suivantes.

$$score\_positif = \sum_{i=1}^{9,\ step=2} (valeur_i - 1)$$

$$score\_negatif = \sum_{i=2}^{10,\ step=2} (5 - valeur_i)$$

$$score\_utilisabilite = (score\_positif + score\_negatif) \times 2,5$$

Le questionnaire SUS typique que nous avons appliqué est détaillé en Annexe C. Pour rester autant que possible fidèles au protocole d'évaluation de SUS, nous avons utilisé le questionnaire originel en langue anglaise. Pour rendre le score d'utilisabilité encore plus interpretable, Bangor et al. (2009) ont établi une correspondance entre ce score et des qualificatifs. Pour ce faire, ils ont réalisé une enquête permettant d'obtenir le score d'utilisabilité de 1000 applications, ainsi qu'un qualificatif que l'utilisateur attribue. À partir de là, ils ont établi une corrélation entre les qualificatifs et les scores moyens d'utilisabilité correspondants (Table 9.4).

TABLE 9.4 – Correspondances entre scores d'utilisabilité et qualificatifs

Qualification	**Score moyen**
Pire possible (*Worse imaginable*)	12,5
Horrible (*Awful*)	20,3
Médiocre (*Poor*)	35,7
Acceptable (*OK*)	50,9
Bien (*Good*)	71,4
Excellent (*Excellent*)	85,5
Meilleur possible (*Best imaginable*)	90,9

Gain de temps et plus-value informationnelle

Les critères de gain de temps et plus-value informationnelle ont été introduits dans le contexte des lacs de données par FERNANDEZ et al. (2018) pour évaluer leur implémentation de lac.

La notion de gain de temps désigne la différence entre le temps requis pour réaliser une tâche à l'aide d'une application et le temps requis pour la même tâche dans une approche « traditionnelle », soit l'utilisation des outils et techniques précédemment connus et exploités par l'utilisateur pour cette tâche.

La notion de plus-value informationnelle vise quant à elle à mesurer la quantité de connaissances désormais accessibles grâce à l'outil, toujours par rapport à une approche dite traditionnelle.

Pour mesurer le gain de temps et la plus-value informationnelle sur leur lac de données, FERNANDEZ et al. (2018) ont recueilli l'appréciation des utilisateurs à travers un questionnaire. Ce questionnaire permet d'évaluer chacun des critères en fonction de chacune des fonctionnalités du lac, à travers une appréciation comprise entre 1 (extrêmement négative) et 5 (extrêmement positive).

Nous proposons un questionnaire similaire, qui se décline à travers les principales fonctionnalités du système AUDAL :

1. la recherche par mots clés,
2. le filtrage par catégories,
3. l'extraction des *highlights*,
4. le *scoring* de documents,
5. le calcul des termes les plus fréquents,
6. la comparaison (*clustering*) de groupes de documents,
7. l'analyse de corrélation entre colonnes de tables,
8. la comparaison de n-uplets (*table clustering*).

Le questionnaire utilisé à cet effet est présenté en Annexe C.

9.4.2 Protocole d'évaluation et résultats

Protocole d'évaluation de l'expérience utilisateur avec AUDAL

Pour mesurer l'utilisabilité, le gain de temps et la plus-value informationnelle associés au système AUDAL, nous utilisons un protocole en trois étapes que nous déroulons avec chaque utilisateur du système. Ce protocole a été soumis à six utilisateurs, dont une moitié a le profil de *data scientist* (doctorants et postdoctorants du laboratoire ERIC) et l'autre celui d'utilisateur métiers (chercheuses en Sciences de gestion du laboratoire Coactis).

1. **Initiation de l'utilisateur aux principales fonctionnalités d'AUDAL.** Il s'agit pour l'utilisateur de réaliser de façon dirigée un ensemble de tâches qui reflètent les huit fonctionnalités principales du système. Cette première étape se déroule durant environ 45 minutes.
2. **Utilisation en autonomie des principales fonctionnalités d'AUDAL.** Au cours de cette étape, l'utilisateur reprend l'ensemble des tâches de l'étape n° 1, avec un maximum d'autonomie. Cette deuxième étape se déroule durant environ 30 minutes.
3. **Réponse au questionnaire.** Dans cette dernière étape, l'utilisateur répond à un formulaire de 26 questions (Annexe C) à travers lequel il fournit de façon anonyme son appréciation sur les trois métriques. Cette dernière étape nécessite environ 15 minutes.

Possibles limites et précautions employées

Dans cette section, nous évoquons les imperfections de notre processus d'évaluation de l'expérience utilisateur. Pour chacune d'elles, nous en soulignons les raisons ou les solutions utilisées pour y remédier.

1. **Faible nombre d'utilisateurs.** L'évaluation de l'expérience utilisateur réalisée pour AUDAL se base sur l'appréciation de seulement six utilisateurs. Ce faible nombre d'utilisateurs s'explique notamment par le caractère chronophage du protocole d'évaluation, qui nécessite environ deux heures à la fois pour l'utilisateur et l'administrateur du système. Nous notons que le système AUDAL a tout de même été évalué par plus d'utilisateurs que le lac de données de Fernandez et al. (2018), qui a été évalué par seulement quatre utilisateurs.
2. **Temps insuffisant de familiarisation avec l'application.** Le ressenti de chaque utilisateur est basé sur 1h15 d'interaction avec l'application. Cela paraît insuffisant pour une prise en main complète de l'application par l'utilisateur. Dans ces conditions, l'utilisateur est susceptible de percevoir le système comme « moins utilisable » que s'il avait eu plus de temps. Nous n'avons cependant pas pu procéder autrement, compte tenu du caractère déjà chronophage du protocole d'évaluation dans sa version actuelle et de l'agenda très chargé des utilisateurs potentiels.
3. **Proximité avec les utilisateurs.** Les six utilisateurs qui ont participé au processus d'évaluation d'AUDAL sont tous des collègues de travail. Pour remédier à un possible biais lié à ce fait, et donc rendre leurs appréciations les plus objectives possibles, nous avons anonymisé le questionnaire d'évaluation.

Utilisabilité de l'interface graphique d'AUDAL

En appliquant le protocole d'évaluation présenté précédemment, nous obtenons un score d'utilisabilité moyen de 65 pour AUDAL. En nous référant aux qualificatifs définis par Bangor et al. (2009) (Table 9.4), nous déduisons que notre système est globalement considéré comme « bon » du point de vue de l'utilisabilité, car ayant un score proche du score moyen correspondant à ce qualificatif (Figure 9.15).

De plus, en nous basant sur les travaux de Kortum et Bangor (2013), nous pouvons comparer le score d'utilisabilité d'AUDAL à celui d'applications très utilisées. Nous constatons ainsi que notre proposition est moins utilisable que Microsoft Word [1] (76,2 de score d'utilisabilité), mais plus utilisable que Microsoft Excel [2] (56,5).

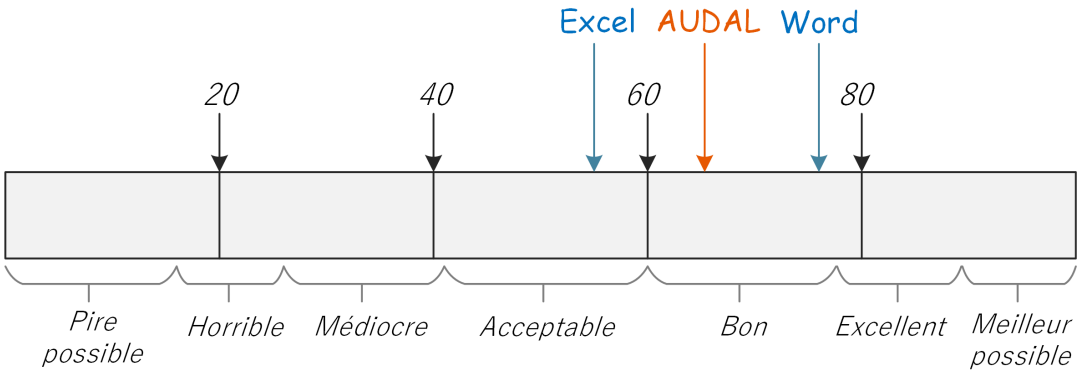


Figure 9.15 – Utilisabilité d'AUDAL par rapport aux qualificatifs de Bangor et al. (2009)

Plus-value informationnelle et gain de temps engendrés par AUDAL

L'évaluation de la plus-value informationnelle et du gain de temps a été déclinée suivant les huit fonctionnalités principales du système AUDAL et effectuée par six utilisateurs. Nous obtenons ainsi un ensemble de $8 \times 6 = 48$ notes comprises entre 1 et 5, pour chacune des deux métriques d'évaluation. Pour dégager une tendance générale, nous proposons d'agréger l'ensemble de ces notes en une distribution représentée sous forme d'histogramme. Nous pouvons ainsi calculer des indicateurs statistiques qui permettent d'analyser les résultats obtenus.

La Figure 9.16 présente la distribution des réponses sur la plus-value informationnelle apportée par AUDAL. Pour rappel, la note minimale (1) exprime une absence totale de plus-value informationnelle, tandis que la note maximale (5) exprime une plus-value absolue. De ce que nous observons, les utilisateurs estiment dans la grande majorité des cas que l'utilisation du système AUDAL permet d'obtenir des informations nouvelles. La moyenne des notes obtenue sur l'évaluation de la plus-value informationnelle est en effet de 3,85, et donc supérieure à la moyenne attendue d'un système « neutre ».

1. https://www.microsoft.com/fr-fr/microsoft-365/word
2. https://www.microsoft.com/fr-fr/microsoft-365/excel

Pour valider statistiquement cette observation, nous comparons la moyenne des notes obtenues à la valeur 3, de sorte à vérifier que cette appréciation positive globale est significative. Pour ce faire, nous appliquons un test de comparaison de moyennes de Student (Livingston, 2004). Il s'agit d'un test statistique permettant de comparer une moyenne observée sur une population (les notes, dans notre cas) à une valeur théorique (dans notre cas, la valeur 3, qui représente le centre de l'intervalle de notes possibles). Les hypothèses nulle H_0 et alternative H_1 de ce test sont formulées comme suit.

— H_0 : la moyenne des notes est égale à 3.

— H_1 : la moyenne des notes est significativement supérieure à 3.

La valeur *p-value* obtenue à l'issue du test est de 0,0001, donc inférieure au risque d'erreur standard de 5 %. Par conséquent, nous pouvons rejeter l'hypothèse nulle, et conclure que la moyenne des notes sur la plus-value informationnelle apportée par AUDAL est significativement supérieure à 3.

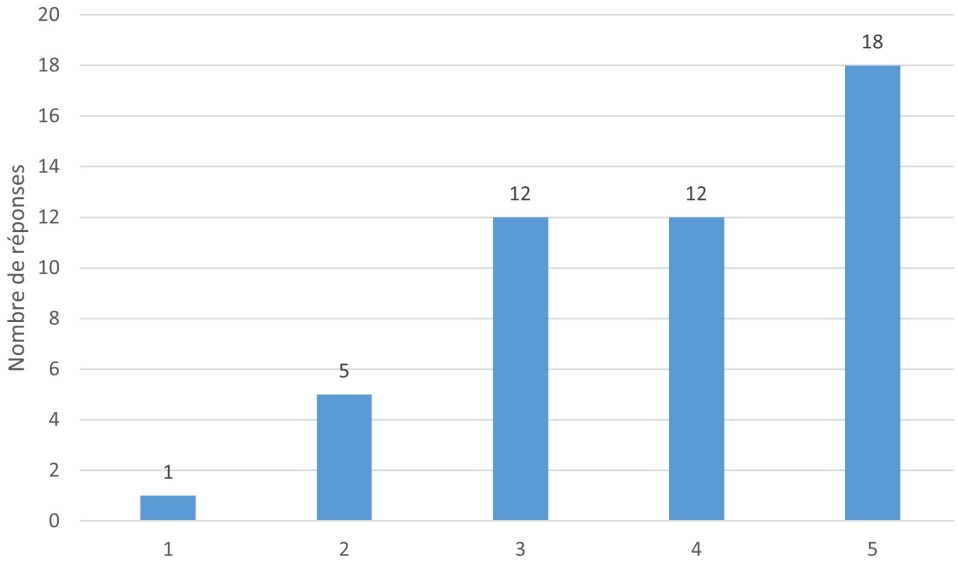


Figure 9.16 – Histogramme des réponses sur la plus-value informationnelle

La Figure 9.17 présente la distribution des réponses sur le gain de temps apporté par AUDAL. D'après nos observations, nous pouvons déduire que le système AUDAL engendre un gain de temps important pour les utilisateurs. Cette conclusion est statistiquement confirmée par le test de Student, qui montre que la moyenne des notes est significativement supérieure à 3. Le résultat de ce test produit en effet une *p-value* de 0,0001, inférieure au risque d'erreur standard de 5 %. On peut donc rejeter l'hypothèse nulle, et affirmer que l'appréciation moyenne du gain apporté par AUDAL (4,23) est significativement supérieure à 3 (la moyenne attendue d'un système « standard ») .

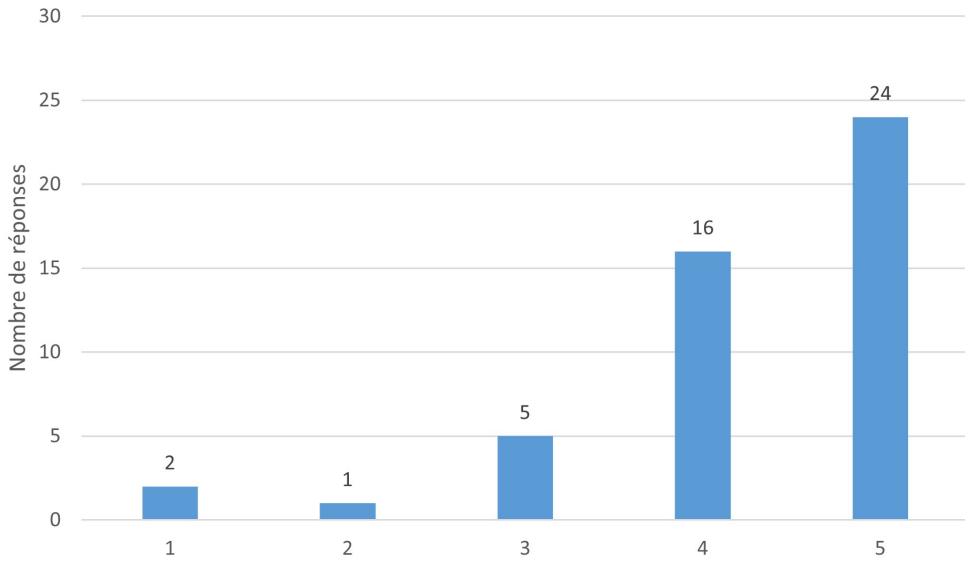


FIGURE 9.17 – Histogramme des réponses sur le gain de temps

9.5 AUDAL face aux exigences de la littérature

Dans cette section, nous proposons une troisième approche d'évaluation du système AUDAL, en le confrontant à des critères définis dans la littérature. Nous considérons, d'une part, les exigences définies par CODD et al. (1993) pour les systèmes OLAP, car AUDAL est assimilable à un système OLAP, surtout sa composante textuelle. D'autre part, nous considérons des exigences plus récentes définies par FABER et al. (2019) pour les systèmes de visualisation de données métier.

9.5.1 AUDAL face aux exigences de Codd

AUDAL, surtout dans sa partie textuelle, satisfait l'ensemble des douze exigences de Codd que nous avons présentées dans la Section 2.2.2. Dans ce qui suit, nous détaillons comment chaque exigence est prise en compte.

1. **Conception multidimensionnelle.** AUDAL répond à cette exigence à travers le concept de groupements. Les documents textuels (et potentiellement les documents tabulaires que nous n'avons pas inclus dans les groupements du fait de leur faible nombre) sont catégorisés en collections organisées qui jouent exactement le même rôle que les dimensions dans les outils OLAP classiques, à la différence que ces dimensions sont plus facilement extensibles dans AUDAL.
2. **Transparence.** AUDAL répond à l'exigence de transparence en permettant d'obtenir des informations détaillées sur les données analysées. L'utilisateur peut ainsi retrouver les titres des documents, les dates de création et de dernière modification et même, dans certains cas, le nom de l'auteur. De plus, contrairement à certains outils OLAP, AUDAL maintient un accès au document sous sa forme brute originelle.
3. **Accessibilité.** L'interface d'analyse d'AUDAL est structurée en trois sections. Une section de filtrage est dédiée aux opérations de type *Slice* & *Dice*. La deuxième sec-

tion permet de définir et de paramétrer le type d'analyse (ou d'agrégation), tandis que la troisième section est dédiée à la visualisation des résultats. Cette claire séparation des fonctionnalités permet une prise en main relativement simple de l'outil, comme en témoigne le score d'utilisabilité obtenu (Section 9.4.2).

4. **Performances.** Comme nous l'avons montré dans la Section 9.3, l'implémentation d'AUDAL offre des performances satisfaisantes et passe globalement bien à l'échelle. Bien qu'une partie des services proposés requiert des temps de traitement importants, cela ne représente qu'une minorité des fonctionnalités. D'ailleurs, des solutions ont été identifiées pour y remédier, notamment une augmentation des capacités de l'infrastructure physique qui héberge le système AUDAL.
5. **Client-Serveur.** AUDAL a été conçu sous la forme d'une application web et suit donc une architecture 3-tiers, ce qui permet un accès simultané à plusieurs utilisateurs. Cela permet même, dans certains cas, à un même utilisateur d'accéder à l'application à travers deux fenêtres, dans le but de comparer des perspectives d'analyse différentes.
6. **Dimensions génériques.** Les groupements proposés par AUDAL s'utilisent tous de la même manière. D'une part, ils servent à filtrer les documents appartenant (ou non) à un ou plusieurs groupes. D'autre part, ils servent d'axes d'agrégation. Par exemple, l'outil d'analyse permet d'agréger les scores des documents textuels suivant les groupes associés à un groupement choisi par l'utilisateur.
7. **Gestion de l'éparsité.** L'utilisation d'un SGBD orienté graphes (Neo4j) dans AUDAL lui permet d'optimiser le stockage des relations entre les documents. Au delà de cette optimisation interne à Neo4J, nous limitons le stockage des relations de similarité en conservant uniquement les plus importantes, ceci dans le but d'assurer une croissance soutenable des métadonnées inter-entités.
8. **Multi-utilisateurs.** AUDAL étant implémenté sous la forme d'une application web, plusieurs utilisateurs peuvent y accéder de façon simultanée sans que cela n'engendre des résultats incohérents.
9. **Croisement de dimensions.** Le filtrage des documents dans AUDAL peut se faire en exploitant et en croisant tous les groupes présents, indépendamment des groupements auxquels ils sont associés.
10. **Manipulation intuitive des données.** L'interface d'analyse d'AUDAL a été conçue à destination des utilisateurs métiers. À ce titre, nous avons adopté une structure qui en facilite l'utilisation. Là encore, cela se voit à travers le score d'utilisabilité obtenu (Section 9.4.2).
11. **Flexibilité.** Dans AUDAL, les groupements permettent de réaliser des agrégations selon plusieurs axes d'analyse, en comparant des groupes issus d'un groupement défini. On peut par exemple comparer les résultats en fonction des entreprises auxquelles les documents sont associés, en fonction du type de document, de la langue dans laquelle le document est rédigé, etc.
12. **Dimensions et agrégations illimitées.** Les groupements exploités par AUDAL sont issus du système de métadonnées et peuvent à ce titre être enrichis par une simple extension du processus de génération des métadonnées. Ils sont donc (théoriquement) illimités. De même, les approches d'agrégation qui coexistent dans AU-

DAL peuvent être facilement enrichies avec des analyses non initialement envisagées, puisque les données restent accessibles sous leur forme brute.

9.5.2 AUDAL face aux exigences de Faber

Faber et al. (2019) proposent un ensemble de neuf exigences auxquelles les systèmes d'analyse de données métiers doivent se conformer. Dans cette section, nous présentons ces exigences et détaillons comment chaucune d'elles est supportée par AUDAL.

1. **Matérialisation des relations.** *Le système doit présenter et caractériser les relations entre les données.*

   Dans AUDAL, la caractérisation s'effectue à travers les analyses de type KMeans et ACP, qui permettent de comparer des groupes de documents textuels sur la base de leurs vocabulaires, ou des n-uplets de documents tabulaires à travers leurs valeurs numériques. Les relations entre les documents sont également présentées à travers les relations de similarités.

2. **Visualisations interactives.** *Cela consiste à inclure des éléments interactifs qui permettent aux utilisateurs de varier les angles de vue.*

   Dans AUDAL, l'interface d'analyse inclut effectivement des listes de choix, des cases à cocher, des onglets et des champs de textes qui permettent aux utilisateurs de paramétrer les analyses à volonté.

3. **Multiplicité des perspectives d'analyses.** *Le système doit proposer une diversité de perspectives d'analyses.*

   Comme nous l'avons expliqué dans la Section 9.5.1, AUDAL exploite les groupements en tant qu'axes d'agrégation des documents textuels. Il inclut ainsi autant de perspectives d'analyses qu'il y a de groupements.

4. **Adaptativité aux évolutions des données.** *L'outil doit s'adapter continuellement aux changements qui interviennent dans la structure des données.*

   Le système de métadonnées sur lequel repose AUDAL a été conçu de sorte à intégrer facilement et aisément de nouveaux documents, groupements et même de nouvelles propriétés (métadonnées) qui sont automatiquement prises en compte par le système.

5. **Intuitivité et facilité d'utilisation.** *La prise en main et l'utilisation du système doivent être simples et aisées*

   Comme nous l'avons mentionné dans la Section 9.5.1, l'interface d'analyse d'AUDAL est structurée de sorte à en faciliter l'utilisation. Le score d'utilisabilité obtenu montre que la facilitation recherchée est (au moins en partie) effective.

6. **Support de données semi et non structurées.** *L'outil d'analyse doit prendre en charge (entre autres) des données semi-structurées et non structurées.*

   Le système AUDAL répond en partie à cette exigence, en incluant des données non structurées que sont les documents textuels. Les données semi-structurées ne sont pas prises en charge par AUDAL pour l'instant, mais pourraient assez facilement être inclues dans une version future.

7. **Accessibilité continue des données.** *Les utilisateurs doivent conserver l'accès aux données pendant les opérations de modifications sur les données.*

   Dans AUDAL, le processus de génération des métadonnées s'exécute en arrière plan, sans interférence avec l'interface d'analyse. Par conséquent, les utilisateurs bénéficient d'un accès continu aux données.

8. **Multiplicité de profils d'utilisateurs.** *Le système doit s'adapter aux profils et expertises des utilisateurs.*

   AUDAL prévoit un accès différencié aux données et métadonnées du lac, selon les compétences de l'utilisateur. Une API REST est dédiée aux *data scientists*, tandis qu'une interface graphique est proposée aux utilisateurs métiers. L'API REST offre plus de flexibilité dans l'analyse et la présentation des données, là où l'interface graphique procure un accès assisté, mais relativement rigide.

9. **Agilité du schéma des données.** *La structure des données doit être dynamique, et non pas fixée d'avance.*

   Dans AUDAL, le principe de polymorphisme permet d'enrichir les données avec de nouveaux schémas (représentations), simplement en modifiant le processus de génération des métadonnées.

En résumé, le système AUDAL répond à huit des neuf exigences définies par Faber et al. (2019). Seule l'exigence d'inclusion de données semi et non structurées (exigence n° 6) n'est pas entièrement prise en charge.

9.6 Conclusion

Dans ce chapitre, nous avons procédé à une évaluation du système AUDAL sous trois angles à savoir les performances (Sections 9.2 et 9.3), l'expérience utilisateur (Section 9.4) et la conformité à des exigences applicables aux systèmes analogues (Section 9.5).

L'évaluation des performances a été réalisée à l'aide du banc d'essais DLBench. Les résultats de cette évaluation montrent qu'AUDAL passe bien à l'échelle en termes de temps de réponse aux requêtes. Ces temps de réponses sont en effet pour la plupart constants ou évoluent linéairement avec le facteur d'échelle. Sur le plan du stockage des métadonnées, l'évaluation illustre le volume important de métadonnées associé à AUDAL, qui dans certains cas représente la moitié des données brutes. La taille des métadonnées évolue toutefois de façon linéaire, voire logarithmique. Tel n'est pas le cas du temps nécessaire pour la génération de ces métadonnées qui, lui, évolue de façon quasi-exponentielle. Ceci est principalement dû aux métadonnées inter-entités, qui nécessitent un très grand nombre d'opérations pour être générées.

Concernant l'expérience utilisateur, une évaluation basée sur le protocole SUS attribue un score d'utilisabilité de 65 à AUDAL. Cela est certes moins bien que d'autres implémentations de lacs de données comme celle de Bagozi et al. (2019), mais tout de même mieux que certains outils communément utilisés pour l'analyse de données comme Excel. L'évaluation de l'expérience utilisateur montre en outre qu'AUDAL apporte aux utilisateurs des informations nouvelles sur les données, en plus d'engendrer un gain de temps par rapport aux outils traditionnellement utilisés.

Enfin, en confrontant notre implémentation à des exigences de référence dans la littérature, nous constatons que le système AUDAL respecte l'ensemble des douze règles définis par Codd et al. (1993) pour les systèmes OLAP, ainsi que huit des neufs exigences de Faber et al. (2019) pour les outils de visualisation de données métiers.

Chapitre 10

Conclusion générale

10.1 Bilan et contributions

Cette thèse avait pour objectif de remédier à une double problématique. La première problématique, qui est d'ordre métier, consistait à proposer un outil évolutif permettant à des utilisateurs métiers d'analyser de façon assistée des documents textuels et tabulaires. En traitant cette première problématique, nous avons dû aborder une deuxième problématique plus générale et d'ordre scientifique en contribuant à la littérature sur le concept de lacs de données. Nos contributions peuvent être regroupées en trois catégories.

10.1.1 Contributions à la désambiguïsation du concept de lac de données

Le concept de lac de données est né en 2010 (Dixon, 2010) et n'est abordé dans la littérature scientifique que depuis 2014, avec les travaux d'O'Leary (2014), d'après Couto et al. (2019). Les lacs de données étaient donc relativement nouveaux dans la littérature académique au début de notre thèse, en 2018. Ils étaient encore souvent perçus comme un simple concept marketing, ou encore systématiquement associés à la technologie Apache Hadoop. Pour remédier à cela, nous avons contribué à désambiguïser le concept de lac de données à travers un travail d'état de l'art, qui a été publié sous la forme d'un article de revue (Sawadogo & Darmont, 2021). Cette revue de la littérature est subdivisé en trois parties.

Définition du concept de lac de données

Nous avons proposé une nouvelle définition des lacs de données, qui se distingue des définitions précédentes à travers l'inclusion (ou la non-exclusion) des utilisateurs métiers parmi les exploitants potentiels des lacs de données. Nous avons complété cette définition avec une comparaison des lacs de données avec les entrepôts de données à travers les stratégies d'ingestion des données d'une part, et les types d'analyse d'autre part. Nous avons également proposé une analyse des forces et défis relatifs aux lacs de données.

Analyse des approches d'organisation des métadonnées

En plus d'une typologie des métadonnées communément inclues dans les lacs de données, nous avons proposé un inventaire des principales méthodes de gestion des métadonnées dans les lacs de données. Nous avons ainsi détaillé cinq *systèmes* de métadonnées, ainsi que cinq autres *modèles* de métadonnées représentatifs de la diversité des pratiques actuelles.

Inventaire des méthodes, architectures et technologies

En complément des approches d'organisation des métadonnées, nous avons proposé une typologie des approches architecturales utilisées dans les lacs de données. Nous avons ainsi distingué des structures de lacs de données en zones, des approches fonctionnelles et hybrides. Nous avons également fait l'inventaire des technologies permettant de prendre en charge les fonctions de stockage, de traitement et d'accès aux données dans les implémentations de lacs de données.

10.1.2 Contributions à la modélisation des métadonnées

Au cours du travail d'état de l'art que nous avons détaillé dans la Section 10.1.1, nous avons constaté des limites aux approches existantes d'organisation des métadonnées dans les lacs de données. La plupart de ces approches étaient en effet spécifiques à des cas d'usage précis et donc difficilement réutilisables. Les seules approches tendant à la généricité, que nous appelons modèles de métadonnées, étaient elles aussi limitées, non seulement par rapport aux types de données pris en charge, mais aussi en termes de fonctionnalités supportées. Nous avons alors contribué à remédier à ces insuffisances à travers deux modèles de métadonnées.

MEDAL, modèle pour l'organisation des métadonnées

MEDAL s'appuie sur une vision étendue des métadonnées, ainsi qu'une typologie en métadonnées intra-entité, inter-entités et globales. Pour positionner MEDAL, nous avons introduit et utilisé un ensemble de six fonctionnalités récurrentes et pertinentes dans la gestion des métadonnées de lacs de données. Une comparaison de MEDAL aux modèles et systèmes de métadonnées pré-existants a ainsi montré que notre proposition était la plus générique à ce moment. La description de MEDAL a été publiée sous la forme d'un article en atelier international (Sawadogo et al., 2019b) et d'un article de conférence nationale (Scholly et al., 2019).

goldMEDAL, modèle générique de métadonnées

Après la publication de MEDAL, plusieurs travaux ont abordé la problématique de la modélisation générique des métadonnées dans les lacs de données. Avec ces travaux, les fonctionnalités attendues d'un modèle générique de métadonnées ont été étendues et une tendance à l'abstraction des concepts de métadonnées a émergé. Pour répondre à ces nouvelles exigences, nous avons proposé le modèle goldMEDAL, une évolution substantielle de MEDAL. Ce nouveau modèle se distingue de son prédécesseur en adoptant un plus haut niveau d'abstraction. Nous avons évalué goldMEDAL en montrant qu'il est capable de modéliser les fonctionnalités de tous les autres modèles, tout en en proposant de nouvelles. La description de goldMEDAL a été publiée sous la forme d'un article d'atelier international qui est la référence en informatique décisionnelle (Scholly et al., 2021b) et soumis en version étendue à la revue internationale *Information Systems*.

10.1.3 Contributions à la mise en œuvre de lacs de données

La modélisation des métadonnées fait partie de la conception théorique du système de métadonnées et, ce faisant, des lacs de données. Elle ne répond cependant pas aux interrogations sur la mise en œuvre effective d'un lac de données. C'est pourquoi nous avons proposé dans un troisième temps un ensemble de contributions portant sur l'implémentation de lacs de données.

AUDAL, lac de données textuelles et tabulaires

À travers le système AUDAL, nous avons proposé une approche de mise en œuvre de lacs de données prenant à la fois en charge des données textuelles et tabulaires. AUDAL

s'appuie sur le modèle MEDAL, ainsi que sur des technologies libres et pour la plupart distribuables sur plusieurs machines (SQLite, MongoDB, Neo4j et Elasticsearch). Cela lui permet de prendre en charge une large panoplie d'analyses allant de la recherche de données à l'analyse de contenus textuels et tabulaires. Ces analyses sont proposées sous la forme de services accessibles via une API REST dédiée aux *data scientists*, d'une part, et à travers une interface graphique destinée aux utilisateurs métiers, d'autre part. Les spécifications d'AUDAL ont été acceptées pour publication sous la forme d'un article de conférence internationale (Sawadogo et al., 2021b).

DLBench, banc d'essais pour l'évaluation quantitative de lacs de données

Le concept de lac de données étant relativement nouveau, il n'existe pas encore de critères partagés pour l'évaluation quantitative des performances des lacs de données. Pour y remédier, nous avons proposé le banc d'essais DLBench, qui est constitué d'un modèle de données textuelles et tabulaires, ainsi que d'un modèle de charge composé de tâches et de métriques adaptées au contexte des lacs de données. Nous avons évalué DLBench en montrant qu'il se conformait aux caractéristique d'un « bon » banc d'essais. Les spécifications de DLBench ont été acceptées pour publication sous la forme d'un article de conférence internationale (Sawadogo et al., 2021a).

Évaluation de l'expérience utilisateur

Nous avons évalué quantitativement les performances d'AUDAL à l'aide de DLBench. Les résultats montrent qu'AUDAL passe globalement bien à l'échelle, mais cela ne renseigne pas sur l'adéquation du système aux besoins des utilisateurs. C'est pourquoi nous avons proposé en plus une évaluation de l'expérience utilisateur. Pour cela, nous avons inventorié et réutilisé les méthodes d'évaluation de l'expérience utilisateur adoptées dans la littérature sur les lacs de données. Au delà de compléter l'évaluation d'AUDAL, cet inventaire offre une vue d'ensemble sur les techniques de mesure de l'expérience utilisateur dans le contexte des lacs de données.

10.2 Perspectives de recherche

Le travail de recherche que nous avons mené au cours de cette thèse nous a permis d'apporter plusieurs contributions à la littérature sur les lacs de données et de répondre à plusieurs interrogations (Section 10.1). Toutefois, ces réponses ont à leur tour ouvert de nouvelles pistes de recherche qui pourraient faire l'objet de travaux futurs. Dans cette section, nous présentons six perspectives de recherche découlant de cette thèse.

10.2.1 Modéliser les métadonnées par des méthodes ensemblistes

Au cours de notre travail d'état de l'art, nous avons souligné l'existence des méthodes ensemblistes pour la modélisation de systèmes décisionnels agiles : les *data vaults* et l'*anchor modeling* (Section 2.3.2). La modélisation en *data vault* a été adoptée par Nogueira et al. (2018) pour organiser les métadonnées d'un lac de données. À travers leur implémentation, les auteurs montrent en effet que cette approche est suffisamment flexible pour s'adapter

au principe d'absence de schéma strict propre aux systèmes de métadonnées des lacs de données. La question reste posée concernant l'approche *anchor modeling*. C'est pourquoi nous envisageons une étude comparative de l'application de ces deux techniques pour le support des métadonnées d'un lac de données. Nous pourrions aussi, de façon plus globale comparer les performances induites par la modélisation ensembliste aux performances des modèles par graphes.

10.2.2 Améliorer la génération des métadonnées

AUDAL comprend un ensemble de processus permettant de générer automatiquement les métadonnées à partir des données brutes. Ces processus nécessitent toutefois d'être améliorés, notamment le processus de génération de métadonnées inter-entités, dont le temps d'exécution évolue de manière exponentielle avec le volume des données brutes. Pour remédier à ce problème, nous envisageons d'étendre la parallélisation des traitements déjà réalisée sur la machine principale en une parallélisation plus large impliquant plusieurs machines.

Une autre solution serait d'utiliser un langage de programmation plus « bas niveau » et plus optimisé comme les langages C ou C++, en lieu et place de Python. Nous envisageons également d'améliorer la qualité des représentations de documents textuels en exploitant des techniques alternatives de vectorisation de documents. Nous pourrions par exemple appliquer des techniques et stratégies d'*embedding* plus adaptées aux longs documents.

10.2.3 Améliorer le temps de réponse des requêtes

Les performances quantitatives du système AUDAL sont globalement satisfaisantes. Elles peuvent nonobstant être encore améliorées, notamment la comparaison de groupes de documents textuels et l'extraction des mots les plus fréquents. Pour y parvenir, nous envisageons d'utiliser des technologies de stockage alternatives. Nous pourrions ainsi remplacer Neo4J par le système orienté graphe JanusGraph[1], MongoDB par un système orienté clés-valeurs comme HBase ou encore Elasticsearch par son concurrent Solr.

10.2.4 Prendre en charge de plus nombreux types de données

Dans sa version actuelle, AUDAL inclut seulement des données structurées tabulaires et non structurées textuelles. Nous envisageons de compléter le spectre de données prises en charge à travers des documents semi-structurés tels que XML et JSON. Cela nécessiterait d'étendre les processus de génération des métadonnées, ainsi que les services d'analyse proposés, de sorte à répondre aux spécificités de ces nouveaux types de documents. Dans le même ordre d'idée, nous envisageons de proposer un système de lac de données multimédia dédié à l'analyse de données non structurées (images, sons et vidéos) par des utilisateurs métiers.

1. https://janusgraph.org/

10.2.5 Étendre le banc d'essais DLBench

Le jeu de données inclus dans le modèle de données de DLBench peut être considéré comme limité en termes de volume dans un contexte *big data*. Cela est particulièrement vrai pour la partie tabulaire, qui ne contient que 5000 documents. Pour y remédier, nous envisageons d'inclure dans DLBench des documents tabulaires plus nombreux et plus volumineux en lieu et place du jeu de données actuel.

Une autre amélioration que nous entendons apporter à DLBench est la prise en compte de la distribution des mots clés dans la constitution des requêtes de recherche de données. Nous pourrions ainsi analyser les temps de réponses avec des mots clés très fréquents, peu fréquents, voire quasiment inexistants.

10.2.6 Limiter et archiver les données du lac

Les lacs de données tels que conçus actuellement ont vocation à ingérer des données sans limite, et donc croître à l'infini. Cette croissance effrenée nous paraît difficilement supportable à long terme. Certes, les coûts de stockage baissent continuellement, mais cela ne suffira pas de notre avis à supporter éternellement la politique du « tout stocker ». Il nous paraît donc necessaire d'anticiper et de refléchir à des moyens de remédier à la probable future satturation de certains lacs de données.

Des solutions à cette (probable future) problématique pouraient passer par la définition de stratégies de filtrage des données ingérées dans le lac d'une part, et de politiques d'archivage des données stockées d'autre part. Pour le filtrage en amont des données ingérées par le lac, nous pourrions par exemple nous appuyer sur des techniques d'intelligence artificielle, pour prédire l'utilité des données potentielles au vue de l'historique d'utilisation du lac, ou encore du type de données déjà inclues. La définition de stratégies d'archivage pourrait quant-à elle s'inspirer de l'idée de B. Inmon (2016) de dédier une partie du lac aux données non utilisées.

Annexes

A Captures d'écran de l'interface d'analyse d'AUDAL

La Figure 1 illustre la structure de l'interface graphique d'AUDAL. La partie gauche, encadrée en violet (1), est dédiée aux tâches de filtrage et d'exploration. La partie droite, encadrée en orange (3), permet de choisir le type d'analyse ou d'agrégation à appliquer aux documents filtrés. La partie centrale, encadrée en bleu (2), présente le résultat sous la forme d'un tableau ou d'une visualisation.

La Figure 2 présente les panneaux dédiés aux tâches de filtrage, en violet (1), et de navigation, en bleu (2). Le panneau de filtrage permet de définir des mots clés que doivent contenir les documents retenus (champ « + matching ») ou des mots clés qu'ils ne doivent pas contenir (champ « - matching »). Ce panneau permet également de faire une recherche floue, en recherchant aussi les termes syntaxiquement proches des termes de requête (champ « fuzzy search »). Le panneau de navigation permet quant à lui de sélectionner ou dé-sélectionner des groupes de documents en fonction des groupements définis dans le système de métadonnées.

La Figure 3 présente un exemple d'analyse de documents textuels qui consiste à extraire les mots clés les plus fréquents. L'encadré (1), en violet, permet aux utilisateurs de paramétrer l'analyse de façon interactive. Ils peuvent notamment changer de représentation (champ « vocabulary »), afficher plus ou moins de mots clés (champ « terms limit ») ou changer de type de visualisation (pour une visualisation en nuage de mots par exemple). L'encadré (2), en bleu, présente des informations sur l'exécution de la tâche. Enfin, l'encadré (3), en orange, affiche la visualisation choisie par l'utilisateur.

La Figure 4 présente un exemple d'analyse de documents tabulaire qui consiste à réaliser un *clustering* KMeans. L'encadré (1), en violet, illustre la construction de la requête permettant d'extraire les n-uplets à classer. L'utilisateur choisit d'abord un document tabulaire de base (champ « main table »), puis éventuellement un deuxième document tabulaire pour réaliser une jointure (champ « join table »). L'utilisateur peut ensuite définir une agrégation des colonnes numériques en fonction d'une colonne textuelle à choisir. Toutes ces interactions engendrent une requête SQL qui est exécutée par le système. L'encadré (2), en bleu, permet de paramétrer l'analyse à effectuer. L'encadré (3), en orange, présente les résultats de l'analyse. Nous observons dans cet exemple une classification de scores des entreprises agrégés par secteur d'activité.

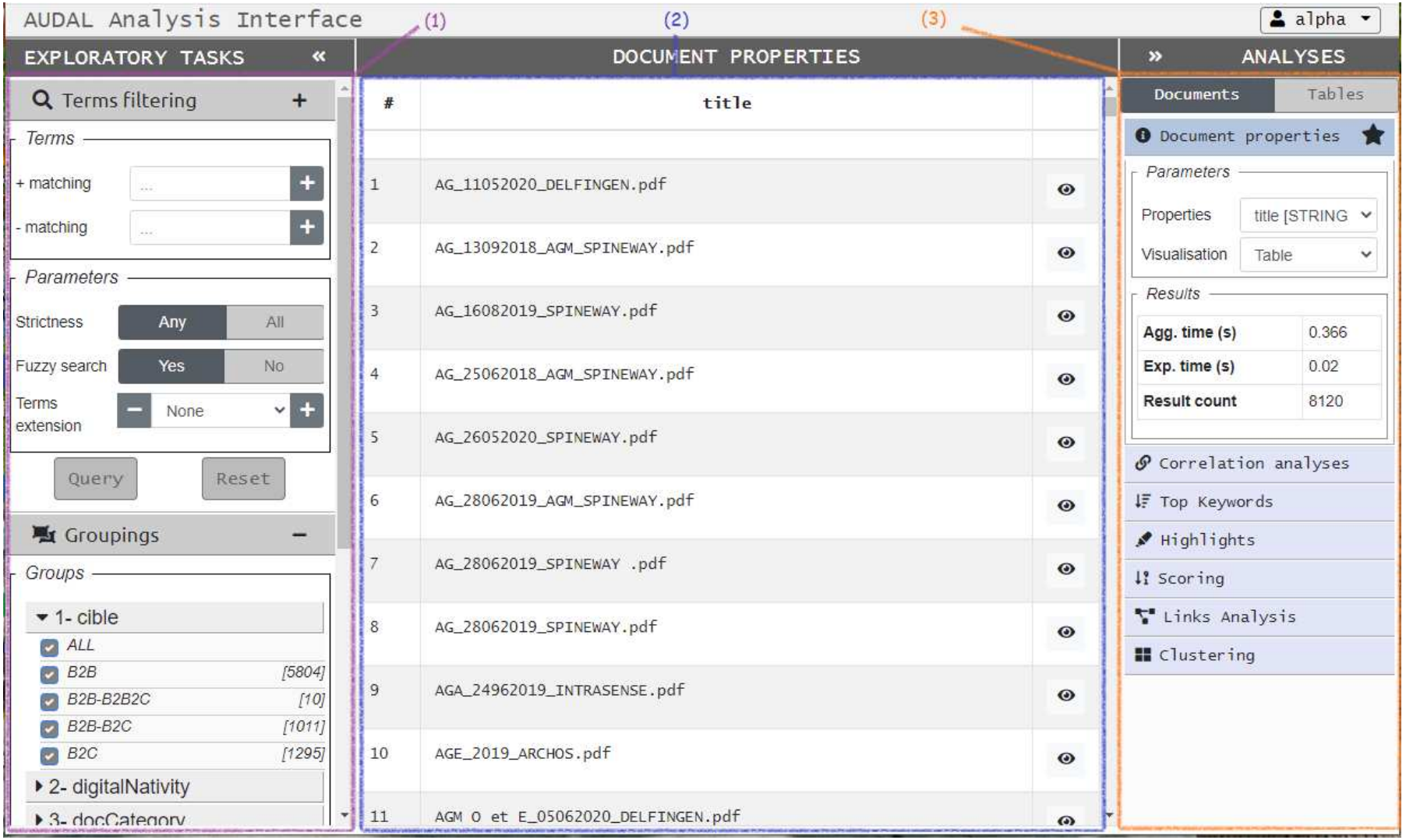


FIGURE 1 – Structure de l'interface graphique d'AUDAL

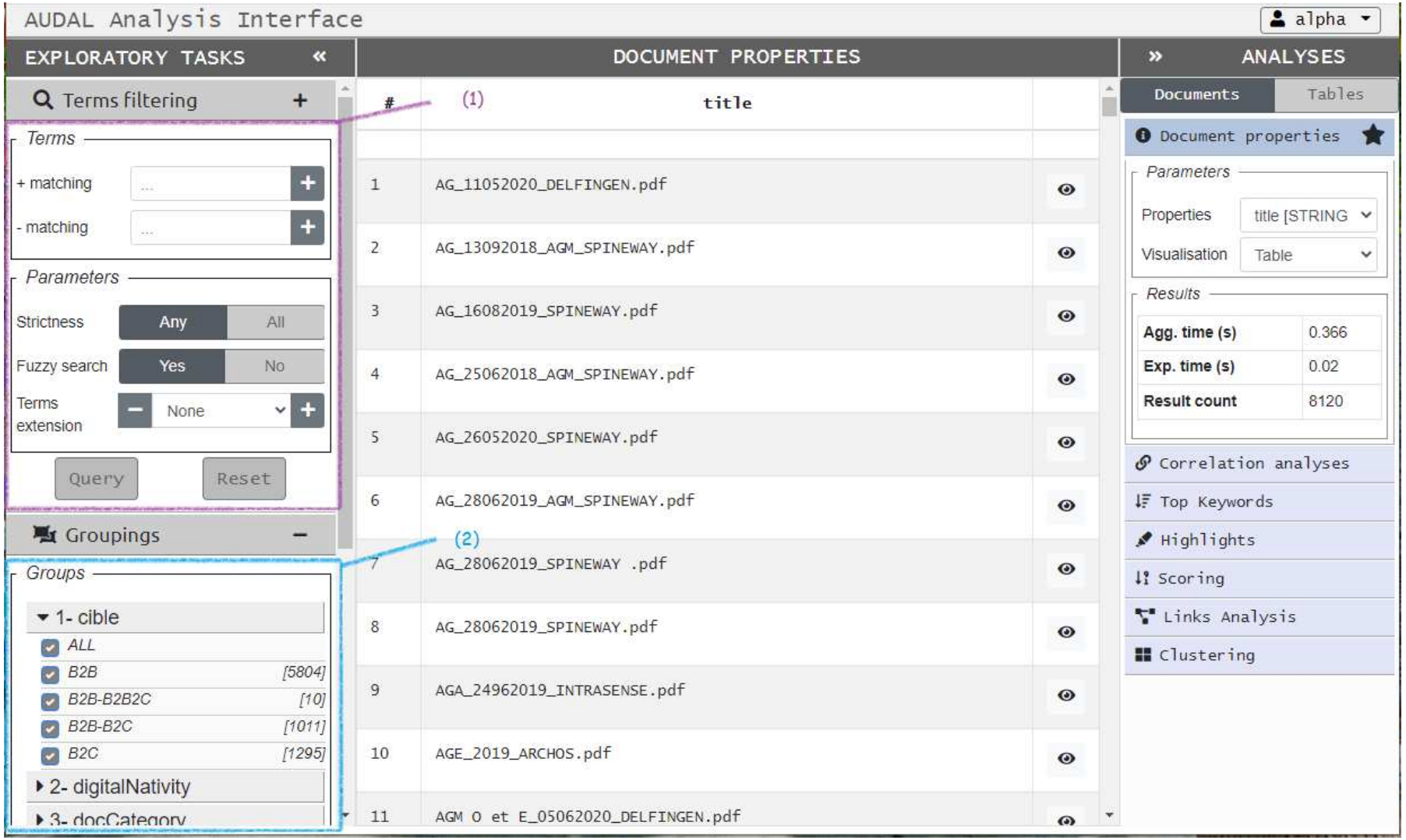


FIGURE 2 – Requêtes de filtrage dans l'interface graphique d'AUDAL

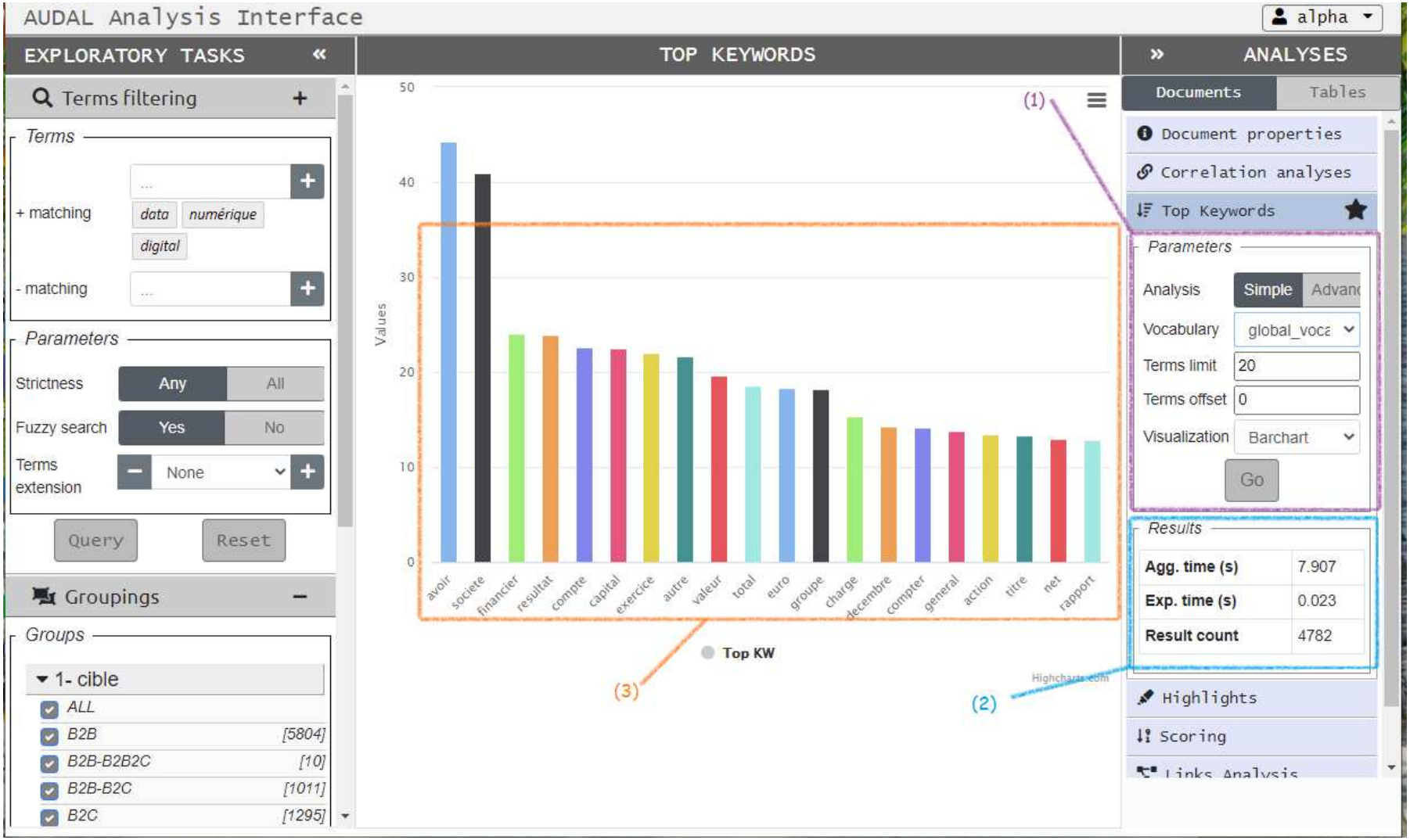


FIGURE 3 – Exemple d'analyse de documents textuels

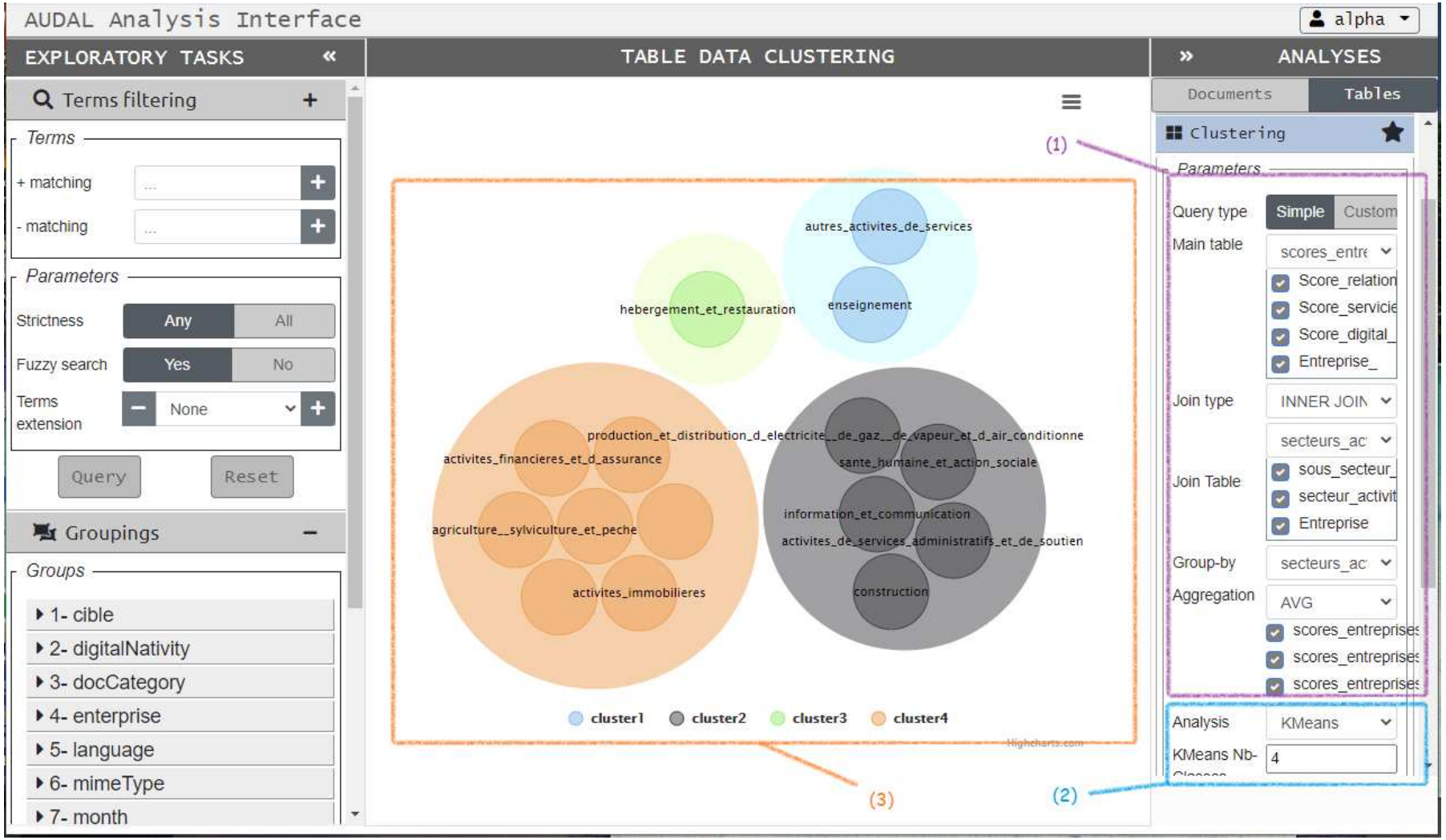


FIGURE 4 – Exemple d'analyse de documents tabulaires

B Protocole d'évaluation de l'expérience utilisateur

Pour vous permettre de réaliser une évaluation objective (dans la mesure du possible), nous vous proposons de vous familiariser avec l'application. Nous proposons ainsi une liste de tâches que vous réaliserez de façon assistée dans un premier temps, et plus en autonomie dans un second temps.

Prise en main des principales fonctionnalités

1. **Recherche par mots clés.** Filtrer les documents contenant les termes data, numérique et digital.
2. **Filtrage par catégories.** En plus de ce premier filtre, faites un filtrage par « catégories », en gardant uniquement les documents dont le docCategory = Presse ou rapports d'activité.
3. **Highlights.** Affichez les endroits d'apparition dans les documents des mots entrés lors du filtrage. Refaites de même avec des fragments de 100 caractères (au lieu de 250 par défaut).
4. **Document Scoring.** Affichez les scores des documents (par rapport aux mots clés entrés lors du filtrage) en les regroupant par docCategory. Faites de même en les regroupant par secteur d'activité.
5. **Top Keywords.** Affichez les mots les plus fréquents en utilisant le vocabulaire aura_pmi. Affichez-le en diagramme en barres et en nuage de mots. Refaites de-même avec le vocabulaire global (global_vocabulary). Affichez le comparatif des termes d'AURA-PMI en fonction du docCategory
6. **Document Clustering.** Faites une présentation K-Means des documents par secteur d'activité, puis par région, en utilisant la représentation aura_pmi. Vous essayerez un K-Means en 2, 3, 4, 5 et 6 classes. Faites une autre présentation ACP (PCA) en suivant les mêmes regroupements et la même représentation.
7. **Table Columns Correlation.** Calculez les corrélations deux à deux entre les scores serviciel, digital et relationnel. Calculez la corrélation entre le score serviciel (table scores_entreprises) et le one_year_beta (table beta_entreprises). Faites d'autres corrélations entre colonnes de scores et colonnes de beta, puis entre colonnes de beta.
8. **Table Clustering.** Faites une présentation ACP des valeurs des scores serviciel, digital et relationnel par rapport au secteur d'activité, puis à la région. Faites de même avec une présentation K-Means à 2, 3, 4, 5 et 6 classes. Affichez la visualisation radar correspondant au K-Means.

Réalisation d'analyses en autonomie

Refaites l'ensemble des 8 analyses ci-dessus, avec cette fois le minimum d'assistance possible.

Évaluation

Vous pouvez à présent proposer votre appréciation de l'outil d'analyse en répondant au questionnaire.

C Questionnaire utilisé pour l'évaluation d'AUDAL

Partie A: Utilisabilité

Dans cette partie, nous proposons un ensemble de 10 questions qui nous permettront d'évaluer à quel point le système est "utilisable" (ou non).

Par souci de fidélité au protocole que nous suivons, nous vous proposons questions dans leur version originale (en anglais).

A1. I think that I would like to use this system frequently

1 (Strongly disagree) ☐
2 ☐
3 ☐
4 ☐
5 (Strongly agree) ☐

A2. I found the system unnecessarily complex

1 (Strongly disagree) ☐
2 ☐
3 ☐
4 ☐
5 (Strongly agree) ☐

A3. I thought the system was easy to use

1 (Strongly disagree) ☐
2 ☐
3 ☐
4 ☐
5 (Strongly agree) ☐

A4. I think that I would need the support of a technical person to be able to use this system

1 (Strongly disagree) ☐
2 ☐
3 ☐
4 ☐
5 (Strongly agree) ☐

A5. I found the various functions in this system were well integrated

1 (Strongly disagree) ☐
2 ☐
3 ☐
4 ☐
5 (Strongly agree) ☐

A6. I thought there was too much inconsistency in this system

1 (Strongly disagree) ☐
2 ☐
3 ☐
4 ☐
5 (Strongly agree) ☐

A7. **I would imagine that most people would learn to use this system very quickly**

1 (Strongly disagree) ☐
2 ☐
3 ☐
4 ☐
5 (Strongly agree) ☐

A8. **I found the system very awkward (hard/difficult) to use**

1 (Strongly disagree) ☐
2 ☐
3 ☐
4 ☐
5 (Strongly agree) ☐

A9. **I felt very confident using the system**

1 (Strongly disagree) ☐
2 ☐
3 ☐
4 ☐
5 (Strongly agree) ☐

A10. **I needed to learn a lot of things before I could get going with this system**

1 (Strongly disagree) ☐
2 ☐
3 ☐
4 ☐
5 (Strongly agree) ☐

Partie B: Plus-value informationnelle (Usefulness)

Dans cette partie, nous vous proposons d'évaluer la valeur des informations fournies par le système à partir des données brutes. Pour cela, nous ciblons six groupes d'analyses que nous estimons représentatives des principales fonctionnalités de l'application.

B1. **Les recherches de documents par "catégories" vous fournissent-elles des informations utiles et/ou nouvelles sur les données (contenu, similarités, ...) ?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

B2. **Les recherches par "mots clés" vous fournissent-elles des informations utiles et/ou nouvelles sur les données (contenu, similarités, ...) ?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

B3. **Les analyses de type "Highlights" vous fournissent-elles des informations utiles et/ou nouvelles sur les données (contenu, similarités, ...) ?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

B4. **Les analyses de type "Document Scoring" vous fournissent-elles des informations utiles et/ou nouvelles sur les données (contenu, similarités, ...) ?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

B5. **Les analyses de type "Top Keywords" vous fournissent-elles des informations utiles et/ou nouvelles sur les données (contenu, similarités, ...) ?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

B6. **Les analyses de type "Document Clustering" vous fournissent-elles des informations utiles et/ou nouvelles sur les données (contenu, similarités, ...) ?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

B7. **Les analyses de type "Table Columns Correlation" vous fournissent-elles des informations utiles et/ou nouvelles sur les données (contenu, similarités, ...) ?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

B8. **Les analyses de type "Table Clustering" vous fournissent-elles des informations utiles et/ou nouvelles sur les données (contenu, similarités, ...) ?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

Partie C: Gain de temps (Time-saving)

Dans cette section, nous vous proposons d'évoluer le gain de temps occasionné par l'utilisation de l'outil AUDAL. Ce gain de temps est à exprimer par rapport aux méthodes que vous utilisez habituellement.

C1. **Les analyses de type "Highlights" vous font-elles gagner du temps par rapport à si vous deviez le faire vous même à partir des données brutes?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

C2. **Les analyses de type "Document Scoring" vous font-elles gagner du temps par rapport à si vous deviez le faire vous même à partir des données brutes?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

C3. **Les analyses de type "Top keywords" vous font-elles gagner du temps par rapport à si vous deviez le faire vous même à partir des données brutes?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

C4. **Les analyses de type "Document Clustering" vous font-elles gagner du temps par rapport à si vous deviez le faire vous même à partir des données brutes?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

C5. **Les analyses de type "Table Columns Correlation" vous font-elles gagner du temps par rapport à si vous deviez le faire vous même à partir des données brutes?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

C6. **Les analyses de type "Table Clustering" vous font-elles gagner du temps par rapport à si vous deviez le faire vous même à partir des données brutes?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

C7. **Les recherches par "mots-clés" vous font-elles gagner du temps par rapport à si vous deviez le faire vous même à partir des données brutes?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

C8. **Les recherches par "catégories" vous font-elles gagner du temps par rapport à si vous deviez le faire vous même à partir des données brutes?**

1 (Pas du tout) ☐
2 ☐
3 ☐
4 ☐
5 (Tout à fait) ☐

Bibliographie

Ajah, I. & Nweke, H. (2019). Big Data and Business Analytics : Trends, Platforms, Success Factors and Applications. *Big Data and Cognitive Computing*, *3*, 32. https://doi.org/10.3390/bdcc3020032

Akhter, A., Ngomo Ngonga, A.-C. & Saleem, M. (2018). An Empirical Evaluation of RDF Graph Partitioning Techniques. In C. Faron Zucker, C. Ghidini, A. Napoli & Y. Toussaint (Éd.), *Knowledge Engineering and Knowledge Management* (p. 3-18). Springer International Publishing. https://doi.org/10.1007/978-3-030-03667-6_1

Allan, J., Lavrenko, V., Malin, D. & Swan, R. (2000). Detections, Bounds, and Timelines : UMass and TDT-3. *Topic Detection and Tracking Workshop (TDT-3), Vienna, VA, USA*, 167-174.

Alrehamy, H. & Walker, C. (2015). Personal Data Lake With Data Gravity Pull. *IEEE 5th International Conference on Big Data and Cloud Computing(BDCloud 2015), Dalian, china*, *88*, 160-167. https://doi.org/10.1109/BDCloud.2015.62

Ansari, J. W., Karim, N., Decker, S., Cochez, M. & Beyan, O. (2018). Extending Data Lake Metadata Management by Semantic Profiling. *2018 Extended Semantic Web Conference (ESWC 2018), Heraklion, Crete, Greece*, 1-15.

Armbrust, M., Ghodsi, A., Xin, R. & Zaharia, M. (2021). Lakehouse : A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics. 11th Annual Conference on Innovative Data Systems Research (CIDR '21). https://cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf

Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A. & Zaharia, M. (2015). Spark SQL : Relational Data Processing in Spark. *Proceedings of the 2015 International Conference on Management of Data (SIGMOD 2015), Melbourne, Victoria, Australia*, 1383-1394. https://doi.org/10.1145/2723372.2742797

Avinoam, R. (2018). ETL Vs ELT : The Difference Is In The How.

Azabou, M., Khrouf, K., Feki, J., Soulé-Dupuy, C. & Vallès, N. (2016). Analyzing Textual Documents with New OLAP Operators. *Proceedings of the 13th International Conference of Computer Systems and Applications (AICCSA 2016), Agadir, Morocco*, 1-8. https://doi.org/10.1109/AICCSA.2016.7945760

Bagozi, A., Bianchini, D., Antonellis, V. D., Garda, M. & Melchiori, M. (2019). Personalised Exploration Graphs on Semantic Data Lakes. *On the Move to Meaningful Internet Systems (OTM 2019), Rhodes, Greece*, 22-39. https://doi.org/$10.1007/978-3-030-33246-4_2$

Bajaber, F., Sakr, S., Batarfi, O., Altalhi, A. H. & Barnawi, A. (2020). Benchmarking big data systems : A survey. *Comput. Commun.*, *149*, 241-251. https://doi.org/10.1016/j.comcom.2019.10.002

Bangor, A., Kortum, P. & Miller, J. (2009). Determining what individual SUS scores mean : Adding an adjective rating scale. *Journal of usability studies*, *4*(3), 114-123.

Beheshti, A., Benatallah, B., Nouri, R., Chhieng, V. M., Xiong, H. & Zhao, X. (2017). CoreDB : a Data Lake Service. *2017 ACM on Conference on Information and Knowledge Management (CIKM 2017), Singapore, Singapore*, 2451-2454. https://doi.org/10.1145/3132847.3133171

Beheshti, A., Benatallah, B., Nouri, R. & Tabebordbar, A. (2018). CoreKG : A Knowledge Lake Service. *Proceedings of the VLDB Endowment*, *11*(12), 1942-1945. https://doi.org/10.14778/3229863.3236230

Bhattacherjee, S. & Deshpande, A. (2018). RStore : A Distributed Multi-Version Document Store. *IEEE 34th International Conference on Data Engineering (ICDE), Paris, France*, 389-400. https://doi.org/10.1109/ICDE.2018.00043

Bogatu, A., Fernandes, A., Paton, N. & Konstantinou, N. (2020). Dataset Discovery in Data Lakes. *36th IEEE International Conference on Data Engineering (ICDE2020), Dallas, Texas, USA*.

Borasi, P., Khan, S. & Kuma, V. (2020). Big Data and Business Analytics Market.

Brewer, E. A. (2000). Towards robust distributed systems.

Brooke, J. (1996). Sus : a quick and dirty usability scale. *Usability evaluation in industry*, *189*.

Bruckner, R. M., List, B. & Schiefer, J. (2002). Striving towards Near Real-Time Data Integration for Data Warehouses. *International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2002)*, 317-326. https://doi.org/10.1007/3-540-46145-0_31

Bruyant, A. & Gateau, D. (1987). Le modèle relationnel pour les données de la production.

Cha, B., Park, S., Kim, J., Pan, S. & Shin, J. (2018). International Network Performance and Security Testing Based on Distributed Abyss Storage Cluster and Draft of Data Lake Framework. *Hindawi Security and Communication Networks*, *2018*, 1-14. https://doi.org/10.1155/2018/1746809

Chanial, C., Dziri, R., Galhardas, H., Leblay, J., Nguyen, M.-H. L. & Manolescu, I. (2018). ConnectionLens : Finding Connections Across Heterogeneous Data Sources. *Proceedings of the VLDB Endowment*, *11*(12), 2030-2033.

Chen, L., Shao, J., Yu, Z., Sun, J., Wu, F. & Zhuang, Y. (2015). RAISE : A Whole Process Modeling Method for Unstructured Data Management. *2015 IEEE International Conference on Multimedia Big Data (BigMM 2015), Beijing, China*, 9-12. https://doi.org/10.1109/BigMM.2015.90

Chessell, M., Scheepers, F., Nguyen, N., van Kessel, R. & van der Starre, R. (2014). *Governing and Managing Big Data for Analytics and Decision Makers*. IBM.

Chockalingam, P. (2019). Open Sourcing Delta Lake.

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, *13*(6), 377-387. https://doi.org/10.1145/362384.362685

Codd, E., Codd, S. & Salley, C. (1993). *Providing OLAP (On-line Analytical Processing) to User-Analysts : An IT Mandate*. E. F. Codd et Associates.

Cohen, W. W., Ravikumar, P., Fienberg, S. E. et al. (2003). A Comparison of String Distance Metrics for Name-Matching Tasks. *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), Acapulco, Mexico*, *3*, 73-78.

Costa, H. (2015). Assessing Comparable Corpora through Distributional Similarity Measures. *EXPERT Scientific and Technological Workshop, Malaga, Spain*, 23-32. https://doi.org/10.13140/RG.2.1.1618.5448

Couto, J., Borges, O., Ruiz, D., Marczak, S. & Prikladnicki, R. (2019). A Mapping Study about Data Lakes : An Improved Definition and Possible Architectures. *31st International Conference on Software Engineering and Knowledge Engineering (SEKE 2019), Lisbon, Portugal*, 453-458. https://doi.org/10.18293/SEKE2019-129

Darmont, J. (2019). Data-Centric Benchmarking. *Advanced Methodologies and Technologies in Network Architecture, Mobile Computing, and Data Analytics* (p. 342-353). IGI Global.

Derakhshannia, M., Gervet, C., Hajj-Hassan, H., Laurent, A. & Martin, A. (2020). Data Lake Governance : Towards a Systemic and Natural Ecosystem Analogy. *Future internet*, *12*(8), 126.

DeWitt, D. J., Halverson, A., Nehme, R. V., Shankar, S., Aguilar-Saborit, J., Avanes, A., Flasza, M. & Gramling, J. (2013). Split query processing in polybase. *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2013), New York, NY, USA*, 1255-1266. https://doi.org/10.1145/2463676.2463709

Diamantini, C., Giudice, P. L., Musarella, L., Potena, D., Storti, E. & Ursino, D. (2018). A New Metadata Model to Uniformly Handle Heterogeneous Data Lake Sources. *New Trends in Databases and Information Systems - ADBIS 2018 Short Papers and Workshop, Budapest, Hungary*, 165-177. https://doi.org/10.1007/978-3-030-00063-9_17

Dixon, J. (2010). Pentaho, Hadoop, and Data Lakes. https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/

Eichler, R., Giebler, C., Gröger, C., Schwarz, H. & Mitschang, B. (2020). HANDLE-A Generic Metadata Model for Data Lakes. *International Conference on Big Data Analytics and Knowledge Discovery (DaWak 2020), Bratislava, Slovakia*, 73-88. https://doi.org/10.1007/978-3-030-59065-9_7

Elasticsearch. (2019). Theory Behind Relevance Scoring. https://www.elastic.co/guide/en/elasticsearch/guide/current/scoring-theory.html

Faber, A., Riemhofer, M., Huth, D. & Matthes, F. (2019). Visualizing Business Ecosystems : Results of a Systematic Mapping Study. *21st International Conference on Enterprise Information Systems (ICEIS 2019), Heraklion, Crete, Greece*, 357-364.

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. *5th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems (CYBER 2015), Shenyang, China*, 820-824. https://doi.org/10.1109/CYBER.2015.7288049

Farid, M., Roatis, A., Ilyas, I. F., Hoffmann, H.-F. & Chu, X. (2016). CLAMS : Bringing Quality to Data Lakes. *2016 International Conference on Management of Data (SIGMOD 2016), San Francisco, CA, USA*, 2089-2092. https://doi.org/10.1145/2882903.2899391

Farrugia, A., Claxton, R. & Thompson, S. (2016). Towards Social Network Analytics for Understanding and Managing Enterprise Data Lakes. *Advances in Social Networks Analysis and Mining (ASONAM 2016), San Francisco, CA, USA*, 1213-1220. https://doi.org/10.1109/ASONAM.2016.7752393

Fauduet, L. & Peyrard, S. (2010). A Data-First Preservation Strategy : Data Management In SPAR. *7th International Conference on Preservation of Digital Objects (iPRES 2010), Vienna, Autria*, 1-8. http://www.ifs.tuwien.ac.at/dp/ipres2010/papers/fauduet-13.pdf

Favre, C. (2007). *Évolution de schémas dans les entrepôts de données : mise à jour de hiérarchies de dimension pour la personnalisation des analyses* (thèse de doct.). Université Lumière-Lyon II.

Fernandez, R. C., Abedjan, Z., Koko, F., Yuan, G., Madden, S. & Stonebraker, M. (2018). Aurum : A Data Discovery System. *34th IEEE International Conference on Data Engineering (ICDE 2018), Paris, France*, 1001-1012.

Fialho, P., Coheur, L. & Quaresma, P. (2020). Benchmarking Natural Language Inference and Semantic Textual Similarity for Portuguese. *Inf.*, *11*(10), 484. https://doi.org/10.3390/info11100484

Ganore, P. (2015). Introduction To The Concept Of Data Lake And Its Benefits.

Gaudiaut, T. (2020). Le big bang du big data.

Ghazal, A., Ivanov, T., Kostamaa, P., Crolotte, A., Voong, R., Al-Kateb, M., Ghazal, W. & Zicari, R. V. (2017). BigBench V2 : The New and Improved BigBench. *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, 1225-1236. https://doi.org/10.1109/ICDE.2017.167

Ghazal, A., Rabl, T., Hu, M., Raab, F., Poess, M., Crolotte, A. & Jacobsen, H. (2013). BigBench : Towards an Industry Standard Benchmark for Big Data Analytics. In K. A. Ross, D. Srivastava & D. Papadias (Éd.), *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013* (p. 1197-1208). ACM. https://doi.org/10.1145/2463676.2463712

Giebler, C., Gröger, C., Hoos, E., Schwarz, H. & Mitschang, B. (2019). Leveraging the Data Lake - Current State and Challenges. *Proceedings of the 21st International Conference on Big Data Analytics and Knowledge Discovery (DaWaK 2019), Linz, Austria.*

Giebler, C., Schwarz, H., Mitschang, B. & und Web, T. (2021). The Data Lake Architecture Framework : A Foundation for Building a Comprehensive Data Lake Architecture. *Datenbanksysteme für Business, Technologie und Web (BTW 2021)*, 351-370. https://doi.org/10.18420/btw2021-19

Gray, J. (1993). Database and Transaction Processing Performance Handbook. http://jimgray.azurewebsites.net/benchmarkhandbook/chapter1.pdf

Grosser, T., Bloeme, J., Mack, M. & Vitsenko, J. (2016). Hadoop and Data Lakes : Use Cases, Benefits and Limitations.

Hai, R., Geisler, S. & Quix, C. (2016). Constance : An Intelligent Data Lake System. *International Conference on Management of Data (SIGMOD 2016), San Francisco, CA, USA*, 2097-2100. https://doi.org/10.1145/2882903.2899389

Hai, R., Quix, C. & Zhou, C. (2018). Query Rewriting for Heterogeneous Data Lakes. *22nd European Conference on Advances in Databases and Information Systems (ADBIS 2018), Budapest, Hungary, 11019*, 35-49. https://doi.org/10.1007/978-3-319-98398-1_3

Halevy, A., Korn, F., Noy, N. F., Olston, C., Polyzotis, N., Roy, S. & Whang, S. E. (2016). Managing Google's data lake : an overview of the GOODS system. *2016 International Conference on Management of Data (SIGMOD 2016), San Francisco, CA, USA*, 795-806. https://doi.org/10.1145/2882903.2903730

Halevy, A. Y., Korn, F., Noy, N. F., Olston, C., Polyzotis, N., Roy, S. & Whang, S. E. (2016). Goods : Organizing Google's Datasets. *Proceedings of the 2016 International Conference on Management of Data (SIGMOD 2016), San Francisco, CA, USA*, 795-806. https://doi.org/10.1145/2882903.2903730

Hamadou, H. B., Ghozzi, F., Péninou, A. & Teste, O. (2018). Querying Heterogeneous Document Stores. *20th International Conference on Enterprise Information Systems (ICEIS 2018), Funchal, Madeira, Portugal*, 58-68.

Haste, J.-L. (2017). From the data lake to the agile data warehouse : decision-making in the big data era.

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. & Sun, E. (2017). Ground : A Data Context Service. *8th Biennial Conference on Innovative Data Systems Research (CIDR 2017), Chaminade, CA, USA*. http://cidrdb.org/cidr2017/papers/p111-hellerstein-cidr17.pdf

Houle, P. (2017). Data Lakes, Data Ponds, and Data Droplets.

Huang, S., Huang, J., Dai, J., Xie, T. & Huang, B. (2010). The HiBench benchmark suite : Characterization of the MapReduce-based data analysis. *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*, 41-51. https://doi.org/10.1109/ICDEW.2010.5452747

Hultgren, H. (2016). *Data Vault modeling guide : Introductory Guide to Data Vault Modeling*. Genessee Academy, USA.

Inmon, B. (2016). *Data Lake Architecture : Designing the Data Lake and avoiding the garbage dump*. Technics Publications.

Inmon, W. H. (1994). *Building the data warehouse*. John Wiley & Sons, Inc.

Inmon, W. H. (2005). *Building the Data Warehouse (Fourth Edition)*. John wiley & sons.

Ivanov, T., Ghazal, A., Crolotte, A., Kostamaa, P. & Ghazal, Y. (2020). CoreBigBench : Benchmarking big data core operations. In P. Tözün & A. Böhm (Éd.), *Proceedings of the 8th International Workshop on Testing Database Systems, DBTest@SIGMOD 2020, Portland, Oregon, June 19, 2020* (4 :1-4 :6). ACM. https://doi.org/10.1145/3395032.3395324

Ivanov, T., Rabl, T., Poess, M., Queralt, A., Poelman, J., Poggi, N. & Buell, J. (2015). Big Data Benchmark Compendium. *Performance Evaluation and Benchmarking : Traditional to Big Data to Internet of Things - 7th TPC Technology*

Conference, TPCTC 2015, Kohala Coast, HI, USA, 135-155. https://doi.org/10.1007/978-3-319-31409-9_9

Ivchenko, G. I. & Honov, S. A. (1998). On the jaccard similarity test. *Journal of Mathematical Sciences*, *88*(6), 789-794. https://doi.org/10.1007/BF02365362

Jain, A. K. (2010). Data clustering : 50 years beyond K-means. *Pattern Recognition Letters*, *31*(8), 651-666. https://doi.org/10.1016/j.patrec.2009.09.011

John, T. & Misra, P. (2017). *Data Lake for Enterprises : Lambda Architecture for building enterprise data systems*. Packt Publishing.

Joss, A. (2016). The Rise of the GDPR Data Lake.

Justel, A., Peña, D. & Zamar, R. (1997). A multivariate Kolmogorov-Smirnov test of goodness of fit. *Statistics & Probability Letters*, *35*(3), 251-259. https://doi.org/https://doi.org/10.1016/S0167-7152(97)00020-5

Khine, P. P. & Wang, Z. S. (2017). Data Lake : A New Ideology in Big Data Era. *4th International Conference on Wireless Communication and Sensor Network (WCSN 2017), Wuhan, China*, *17*, 1-6. https://doi.org/10.1051/itmconf/2018170302

Kimball, R. (1996). *The data warehouse toolkit : practical techniques for building dimensional data warehouses*. John Wiley & Sons, Inc.

Kimball, R. & Ross, M. (2013). *The Data Warehouse Toolkit : The Definitive Guide to Dimensional Modeling*. John Wiley & Sons.

Klettke, M., Awolin, H., Stürl, U., Müller, D. & Scherzinger, S. (2017). Uncovering the Evolution History of Data Lakes. *2017 IEEE International Conference on Big Data (BIGDATA 2017), Boston, MA, USA*, 2462-2471. https://doi.org/10.1109/BigData.2017.8258204

Kolev, B., Bondiombouy, C., Valduriez, P., Jiménez-Peris, R., Pau, R. & Pereira, J. (2016). The CloudMdsQL Multistore System. In F. Özcan, G. Koutrika & S. Madden (Éd.), *Proceedings of the 2016 International Conference on Management of Data (SIGMOD 2016), San Francisco, CA, USA* (p. 2113-2116). ACM. https://doi.org/10.1145/2882903.2899400

Kondrak, G. (2005). N-Gram Similarity and Distance. *International Symposium on String Processing and Information Retrieval (SPIRE 2005)*, 115-126. https://doi.org/10.1007/11575832_13

Kortum, P. T. & Bangor, A. (2013). Usability ratings for everyday products measured with the system usability scale. *International Journal of Human-Computer Interaction*, *29*(2), 67-76.

Krishnan, K. (2013). *Data Warehousing in the Age of Big Data*. Morgan Kaufmann.

Krneta, D., Jovanovic, V. & Marjanovic, Z. (2014). A Direct Approach to Physical Data Vault Design. *Computer Science and Information Systems*, *11*(2), 569-599. https://doi.org/10.2298/CSIS130523034K

Langit, L. (2007). Introduction to MDX. *Foundations of SQL Server 2005 Business Intelligence* (p. 219-242). Apress. https://doi.org/10.1007/978-1-4302-0248-6_10

LaPlante, A. & Sharma, B. (2016). Architecting Data Lakes Data Management Architectures for Advanced Business Use Cases.

Laskowski, N. (2016). Data lake governance : A big data do or die.

Le, Q. & Mikolov, T. (2014). Distributed representations of sentences and documents. *International conference on machine learning*, 1188-1196.

Leclercq, É. & Savonnet, M. (2018). A Tensor Based Data Model for Polystore : An Application to Social Networks Data. *Proceedings of the 22nd International Database Engineering & Applications Symposium (IDEAS 2018), Villa San Giovanni, Italy*, 110-118. https://doi.org/10.1145/3216122.3216152

Lepine, B. (2018). HADOOP - Tout savoir sur la principale plateforme Big Data.

Lewis, J. R. (2018). The system usability scale : past, present, and future. *International Journal of Human–Computer Interaction*, *34*(7), 577-590.

Linstedt, D. (2011). *Super Charge your Data Warehouse : Invaluable Data Modeling Rules to Implement Your Data Vault*. CreateSpace Independent Publishing.

Liu, S., Zhou, M. X., Pan, S., Qian, W., Cai, W. & Lian, X. (2009). Interactive, Topic-Based Visual Text Summarization and Analysis, 543-552. https://doi.org/10.1145/1645953.1646023

Livingston, E. H. (2004). Who was student and why do we care so much about his t-test ? 1. *Journal of Surgical Research*, *118*(1), 58-65.

Maccioni, A. & Torlone, R. (2017). Crossing the finish line faster when paddling the data lake with KAYAK. *VLDB Endowment*, *10*(12), 1853-1856. https://doi.org/10.14778/3137765.3137792

Maccioni, A. & Torlone, R. (2018). KAYAK : A Framework for Just-in-Time Data Preparation in a Data Lake. *International Conference on Advanced Information Systems Engineering (CAiSE 2018), Tallin, Estonia*, 474-489. https://doi.org/$10.1007/978-3-319-91563-0_29$

Madera, C. (2018). *L'évolution des systèmes et architectures d'information sous l'influence des données massives : les lacs de données* (thèse de doct.). Université de Montpellier, France.

Madera, C. & Laurent, A. (2016). The next information architecture evolution : the data lake wave. *8th International Conference on Management of Digital EcoSystems (MEDES 2016), Biarritz, France*, 174-180. https://doi.org/10.1145/3012071.3012077

Malysiak-Mrozek, B., Stabla, M. & Mrozek, D. (2018). Soft and Declarative Fishing of Information in Big Data Lake. *IEEE Transactions on Fuzzy Systems*, *26*(5), 2732-2747. https://doi.org/10.1109/TFUZZ.2018.2812157

Maroto, C. (2018). Data Lake Security – Four Key Areas to Consider When Securing Your Data Lake.

Mathis, C. (2017). Data Lakes. *Datenbank-Spektrum*, *17*(3), 289-293. https://doi.org/10.1007/s13222-017-0272-7

Mehmood, H., Gilman, E., Cortes, M., Kostakos, P., Byrne, A., Valta, K., Tekes, S. & Riekki, J. (2019). Implementing Big Data Lake for Heterogeneous Data Sources. *2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW)*, 37-44. https://doi.org/10.1109/ICDEW.2019.00-37

Messaoud, R. B., Boussaid, O. & Rabaséda, S. (2004). A new OLAP aggregation based on the AHC technique. *Proceedings of the 7th ACM international workshop on Data warehousing and OLAP*, 65-72.

Miloslavskaya, N. & Tolstoy, A. (2016). Big Data, Fast Data and Data Lake Concepts. *7th Annual International Conference on Biologically Inspired Cognitive Architectures (BICA 2016), NY, USA*, *88*, 300-305. https://doi.org/10.1016/j.procs.2016.07.439

Moniruzzaman, A. & Hossain, S. A. (2013). NoSQL Database : New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. *International Journal of Database Theory and Application*, *6*(4).

Nargesian, F., Pu, K. Q., Zhu, E., Bashardoost, B. G. & Miller, R. J. (2018a). Optimizing Organizations for Navigating Data Lakes.

Nargesian, F., Zhu, E., Pu, K. Q. & Miller, R. J. (2018b). Table Union Search on Open Data. *Proceedings of the VLDB Endowment*, *11*, 813-825. https ://arxiv.org/abs/1812.0702 https://doi.org/10.14778/3192965.3192973

Nogueira, I., Romdhane, M. & Darmont, J. (2018). Modeling Data Lake Metadata with a Data Vault. *22nd International Database Engineering and Applications Symposium (IDEAS 2018), Villa San Giovanni, Italia*, 253-261.

O'Leary, D. E. (2014). Embedding AI and Crowdsourcing in the Big Data Lake. *IEEE Intelligent Systems*, *29*(5), 70-73. https://doi.org/10.1109/MIS.2014.82

Ong, K. W., Papakonstantinou, Y. & Vernoux, R. (2014). The SQL++ query language : Configurable, unifying and semi-structured. *arXiv preprint arXiv :1405.3631*.

Oram, A. (2015). *Managing the Data Lake*. Zaloni.

Oukid, L., Boussaid, O., Benblidia, N. & Bentayeb, F. (2016). TLabel : A New OLAP Aggregation Operator in Text Cubes. *IJDWM*, *12*(4), 54-74. https://doi.org/10.4018/IJDWM.2016100103

Parageaud, C. (2013). Big Data : La jungle des différentes distributions open source Hadoop.

Pathirana, N. (2015). *Modeling Industrial and Cultural Heritage Data* (mém. de mast.). Université Lumière Lyon 2, France.

Polyzotis, N., Skiadopoulos, S., Vassiliadis, P., Simitsis, A. & Frantzell, N.-E. (2007). Supporting streaming updates in an active data warehouse. *23rd International Conference on Data Engineering (ICDE 2007)*, 476-485. https://doi.org/10.1109/ICDE.2007.367893

Quix, C. & Hai, R. (2018). Data Lake. *Encyclopedia of Big Data Technologies* (p. 1-8). Springer International Publishing. https://doi.org/$10.1007/978-3-319-63962-8_7-1$

Quix, C., Hai, R. & Vatov, I. (2016). Metadata Extraction and Management in Data Lakes With GEMMS. *Complex Systems Informatics and Modeling Quarterly*, (9), 289-293. https://doi.org/10.7250/csimq.2016-9.04

Ramakrishnan, R., Sridharan, B., Douceur, J. R., Kasturi, P., Krishnamachari-Sampath, B., Krishnamoorthy, K., Li, P., Manu, M., Michaylov, S., Ramos, R., Sharman, N., Xu, Z., Barakat, Y., Douglas, C., Draves, R., Naidu, S. S., Shastry, S., Sikaria, A., Sun, S. & Venkatesan, R. (2017). Azure Data Lake Store : A Hyperscale Distributed File Service for Big Data Analytics. *2017 ACM International Conference on Management of Data (SIGMOD 2017), Chicago, IL, USA*, 51-63. https://doi.org/10.1145/3035918.3056100

Rangarajan, S., Liu, H., Wang, H. & Wang, C.-L. (2018). Scalable Architecture for Personalized Healthcare Service Recommendation Using Big Data Lake. *Australian Symposium on Service Research and Innovation (ASSRI 2017)*, 65-79. https://doi.org/$10.1007/978-3-319-76587-7_5$

Ravat, F., Teste, O. & Tournier, R. (2007). Olap aggregation function for textual data warehouse. *ICEIS (1)*, 151-156.

Ravat, F., Teste, O., Tournier, R. & Zurfluh, G. (2008). Top-keyword : An Aggregation Function for Textual Document OLAP. *10th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2008), Turin, Italy*, 55-64. https://doi.org/$10.1007/978-3-540-85836-2_6$

Ravat, F. & Zhao, Y. (2019a). Data Lakes : Trends and Perspectives. *30th International Conference on Database and Expert Systems Applications (DEXA 2019), Linz, Austria.*

Ravat, F. & Zhao, Y. (2019b). Metadata management for data lakes. *23rd European Conference on Advances in Databases and Information Systems (ADBIS 2019), Bled, Slovenia.*

Regardt, O., Rönnbäck, L., Bergholtz, M., Johannesson, P. & Wohed, P. (2009). Anchor modeling. *International Conference on Conceptual Modeling*, 234-250.

Rehman, N. U., Mansmann, S., Weiler, A. & Scholl, M. H. (2012). Building a data warehouse for twitter stream exploration. *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Istanbul, Turkey*, 1341-1348.

Riley, J. (2017). Understanding Metadata : What is metadata and what is it for ?

Rönnbäck, L. & Hultgren, H. (2013). Comparing Anchor Modeling with Data Vault Modeling. *Modeling for the modern Data Warehouse, White paper.*

Rönnbäck, L., Regardt, O., Bergholtz, M., Johannesson, P. & Wohed, P. (2010). Anchor modeling — Agile information modeling in evolving data environments. *Data & Knowledge Engineering*, *69*(12), 1229-1253. https://doi.org/10.1016/j.datak.2010.10.002

Rosencrance, L. (2019). Les principales distributions Hadoop sur le marché.

Russom, P. (2017). Data Lakes Purposes, Practices, Patterns, and Platforms.

Sawadogo, P. N. & Darmont, J. (2021). On data lake architectures and metadata management. *Journal of Intelligent Information Systems*, *56*(1), 97-120. https://doi.org/10.1007/s10844-020-00608-7

Sawadogo, P. N., Darmont, J. & Noûs, C. (2021a). Benchmarking Data Lakes Featuring Structured and Unstructured Data with DLBench. *Proceedings of the 23rd International Conference on Big Data Analytics and Knowledge Discovery (DaWaK2021), Linz, Austria.*

Sawadogo, P. N., Darmont, J. & Noûs, C. (2021b). Joint Management and Analysis of Textual Documents and Tabular Data within the AUDAL Data Lake. *Proceedings of the 25th European Conference on Advances in Databases and Information Systems (ADBIS 2021), Tartu, Estonia.*

Sawadogo, P. N., Kibata, T. & Darmont, J. (2019a). Metadata Management for Textual Documents in Data Lakes. *21st International Conference on Enterprise Information Systems (ICEIS 2019), Heraklion, Crete, Greece*, 72-83. https://doi.org/10.5220/0007706300720083

Sawadogo, P. N., Scholly, E., Favre, C., Ferey, É., Loudcher, S. & Darmont, J. (2019b). Metadata Systems for Data Lakes : Models and Features. *BI and Big Data Applications - ADBIS 2019 Short Papers and Workshop, Bled, Slovenia.*

Scholly, E., Favre, C., Ferey, E. & Loudcher, S. (2021a). HOUDAL : A Data Lake Implemented for Public Housing. *23rd International Conference on Enterprise In-*

formation Systems (ICEIS 2021) - Volume 1, 39-50. https://doi.org/10.5220/0010418200390050

Scholly, E., Sawadogo, P., Favre, C., Ferey, E., Loudcher, S. & Darmont, J. (2019). Systèmes de métadonnées dans les lacs de données : modélisation et fonctionnalités. *15e journées EDA Business Intelligence & Big Data (EDA 2019)*, 77-92.

Scholly, E., Sawadogo, P. N., Liu, P., Espinosa-Oviedo, J.-A., Favre, C., Loudcher, S., Darmont, J. & Noûs, C. (2021b). 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*, 31-40.

Singh, K., Paneri, K., Pandey, A., Gupta, G., Sharma, G., Agarwal, P. & Shroff, G. (2016). Visual Bayesian Fusion to Navigate a Data Lake. *19th International Conference on Information Fusion (FUSION 2016), Heidelberg, Germany*, 987-994.

Sirosh, J. (2016). The Intelligent Data Lake. https://azure.microsoft.com/fr-fr/blog/the-intelligent-data-lake/

Smith, J. M., Bernstein, P. A., Dayal, U., Goodman, N., Landers, T., Lin, K. W. T. & Wong, E. (1981). Multibase : Integrating Heterogeneous Distributed Database Systems. *Proceedings of the National Computer Conference (NCC 1981), Chicago, Illinois*, 487-499. https://doi.org/10.1145/1500412.1500483

Stefanowski, J., Krawiec, K. & Wrembel, R. (2017). Exploring Complex and Big Data. *International Journal of Applied Mathematics and Computer Science*, *27*(4), 669-679. https://doi.org/10.1515/amcs-2017-0046

Stein, B. & Morrison, A. (2014). The enterprise data lake : Better integration and deeper analytics. http://www.smallake.kr/wp-content/uploads/2017/03/20170313%5C_074222.pdf

Subramaniam, P., Ma, Y., Li, C., Mohanty, I. & Fernandez, R. C. (2021). Comprehensive and Comprehensible Data Catalogs : The What, Who, Where, When, Why, and How of Metadata Management. *CoRR*, *abs/2103.07532*. https://arxiv.org/abs/2103.07532

Sulova, S. (2019). The Usage of Data Lake for Business Intelligence Data Analysis. *Conferences of the department Informatics*, (1), 135-144. https://ideas.repec.org/a/vrn/katinf/y2019i1p135-144.html

Suriarachchi, I. & Plale, B. (2016). Crossing Analytics Systems : A Case for Integrated Provenance in Data Lakes. *12th IEEE International Conference on e-Science (e-Science 2016), Baltimore, MD, USA*, 349-354. https://doi.org/10.1109/eScience.2016.7870919

Tallarida, R. J. & Murray, R. B. (1987). Chi-Square Test. *Manual of Pharmacologic Calculations : With Computer Programs* (p. 140-142). Springer New York. https://doi.org/10.1007/978-1-4612-4974-0_43

Tan, R., Chirkova, R., Gadepally, V. & Mattson, T. G. (2017). Enabling query processing across heterogeneous data models : A survey. *2017 IEEE International Conference on Big Data (Big Data 2017), Boston, MA, USA*, 3211-3220. https://doi.org/10.1109/BigData.2017.8258302

Terrizzano, I., Schwarz, P., Roth, M. & Colino, J. E. (2015). Data Wrangling : The Challenging Journey from the Wild to the Lake. *7th Biennial Conference on Innovative Data Systems Research (CIDR 2015), Asilomar, CA, USA*, 1-9. http://cidrdb.org/cidr2015/Papers/CIDR15%5C_Paper2.pdf

Teste, O. (2000). *Modélisation et manipulation d'entrepôts de données complexes et historisées* (thèse de doct.). Université Paul Sabatier-Toulouse III.

Tharrington, M. (2017). The Dzone Guide to Big Data, Data Science & Advanced Analytics.

Tiao, S. (2018). Object Storage for Big Data : What Is It ? And Why Is It Better ?

TPC. (2014). TPC Benchmark H - Standard Specification (version 2.18.0).

TPC. (2018). TPC Express Benchmark HS - Standard Specification (version 2.0.3).

TPC. (2020a). TPC Benchmark DS - Standard Specification (version 2.13.0).

TPC. (2020b). TPC Express AI - Draft Specification (version 0.6).

Truica, C., Apostol, E. S., Darmont, J. & Assent, I. (2021). TextBenDS : a Generic Textual Data Benchmark for Distributed Systems. *Inf. Syst. Frontiers*, *23*(1), 81-100. https://doi.org/10.1007/s10796-020-09999-y

Wang, L., Zhan, J., Luo, C., Zhu, Y., Yang, Q., He, Y., Gao, W., Jia, Z., Shi, Y., Zhang, S. et al. (2014). Bigdatabench : A big data benchmark suite from internet services. *2014 IEEE 20th international symposium on high performance computer architecture (HPCA)*, 488-499.

Wibowo, M., Sulaiman, S. & Shamsuddin, S. M. (2017). Machine Learning in Data Lake for Combining Data Silos. *International Conference on Data Mining and Big Data (DMBD 2016), Fukuoka, Japan*, *10387*, 294-306. https://doi.org/10.1007/978-3-319-61845-6_30

Wold, S., Esbensen, K. & Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, *2*(1), 37-52. https://doi.org/10.1016/0169-7439(87)80084-9

Zhou, H., Yang, D. & Xu, Y. (2011). An ETL Strategy for Real-Time Data Warehouse. *Practical Applications of Intelligent Systems*, 329-336. https://doi.org/10.1007/978-3-642-25658-5_41

Zhu, Y., Xie, Z., Jin, L., Chen, X., Huang, Y. & Zhang, M. (2019). SCUT-EPT : New Dataset and Benchmark for Offline Chinese Text Recognition in Examination Paper. *IEEE Access*, *7*, 370-382. https://doi.org/10.1109/ACCESS.2018.2885398

Zikopoulos, P., deRoos, D., Bienko, C., Buglio, R. & Andrews, M. (2015). *Big Data Bayond the Hype*. McGraw-Hill Education.