République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université Mohamed Seddik Ben Yahia, Jijel
Faculté des sciences exactes et informatique
Département Informatique

Mémoire de fin d'étude pour obtention du
Diplôme Master de Recherche en Informatique
Option : Systèmes d'Information et Aide à la Décision

Thème

Développement d'un système d'Exploration de Lacs de données NoSQL guidé par les métadonnées

Réalisé par:
- Meriem BOURAOUI
- Chaima GRINE

Encadré par :
- Doulkifli BOUKRAA

Année universitaire : 2021-2022

Remerciements

Tout d’abord et avant tout, nous remercions **DIEU** le tout qui nous a donné la force, la volonté, la patience et le courage pour accomplir ce modeste travail. Merci de nous avoir éclairé sur le chemin de la réussite.

Nous voudrions exprimer toute notre gratitude à notre encadrant **Mr. BOUKRAA Doulkifli**, pour la confiance qu’il nous a accordées en acceptant de nous encadrer, pour sa patience, ses remarques, ses conseils, et surtout sa disponibilité sa bienveillance, et le temps qu’il nous a consacré tout au long de cette période, et répondre à toutes nos interrogations, nous avons appris des choses grâce à lui.

Notre vif remerciement aux membres du jury pour l’honneur qu’ils nous font en acceptant de juger ce travail et pour toutes leurs remarques pertinentes.

Nous n’oublions pas de remercier sincèrement toutes les personnes qui ont participé de près ou de loin à la rédaction de ce mémoire, par leurs conseils et remarques

Dédicace

J’ai l’honneur de dédier le fruit de mes efforts aux plus précieux des trésors :

Mon très cher Père ***Abdelkarim***: aucune dédicace ne saurait exprimer l'amour, l'estime et le respect que j'ai toujours pour toi. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien-être. Ce travail est le fruit de tes sacrifices que vous avez consentis pour ma formation le long de ces années.

Ma tendre Mère ***Samia*** : Tu représentes pour moi la source de tendresse et l'exemple de dévouement qui n'a pas cessé de m'encourager. Tu as fait plus qu'une mère puisse faire pour que ses enfants, tant donné et tant enseigné, toi qui m’as guidé dans le droit chemin, toi qui m’as appris que rien n’est impossible...

À mes belles-sœurs : ***Maria***, ***Bouchra*** et ***Maram*** merci pour tous votre amours, votre tendresse, votre soutien tout au long de mes études.

À mes beaux-frères : ***AbdelGhani***, ***ZineEldinne*** et ***Hamza*** merci pour vos appuis et vos encouragements je vous souhaite un avenir brillant, plein de succès et de bonheur.

À mon binôme ***Meriem***

Je voulais crier haut et fort dédicace à toi ma chérie je remercie vivement pour t'aide et supporte dans les moments difficiles. Je te souhaite une bonne continuation dans ta vie.

À Mes chères copines : ***Amina***, ***Nada***, ***Khadidja*** et ***Yousra*** je tiens à vous exprimer toute ma gratitude, l'amour, le respect et la reconnaissance.

À moi-même, Mlle ***Chaima Grine*** pour le sacrifice que j'ai consenti jour et nuit durant toute ma carrière pour arriver à ce niveau.

À tous ceux qui sont chers qui m'ont soutenu durant toute la durée de réalisation de ce travail et que j’ai omis involontairement de citer.

Chaima

Dédicace

Avec l'expression de ma reconnaissance je dédie ce modeste travail à ceux qui, quels que soient les termes embrassés, je n'arriverais jamais à leur exprimer mon amour sincère

Á mes très chers parents

Mon père Idriss, ma mère Ourida Pour les sacrifices déployés à mon égard, pour leurs soutiens, tous les efforts consentis pour mon éducation et ma formation durant toutes les années d'études et sans lesquels je n'aurais jamais réussi.

Que Dieu tout puissant vous protège, Merci de mon profond cœur et je vous aime.

Á mon adorable sœur

Asma Qui ne cesse de m'encourager et me soutenir tout le temps, Merci et je t'aime.

Á mes très chers frères

Yaakoub, Abderhmane, Chouaib et Soheib qui étaient toujours à mes côtés avec leurs précieux conseils, Merci et je vous aime.

Á mes chers tantes et oncles

Pour leur soutien tout au long de mon parcours universitaire, merci et je vous aime.

Á mes très chers amis

Yousra, Nada, Rima, Dounia, Saliha et Khadija pour m'avoir écouté et réconforté, Merci et je vous aime.

Á ma chère Binôme

Chaima qui a partagé ensemble les bons et mauvais moments de ce travail, Merci et je t'aime.

Á moi-même

Pour ne pas avoir abandonné, je suis très fière de moi !

Á ceux qui sont chers qui m'ont soutenu durant toute la durée de réalisation de ce travail et que j'ai omis involontairement de citer.

Meriem

ملخص

يتطلب استغلال البيانات الضخمة المخزنة في بحيرات البيانات مرحلة من اكتشاف واستكشاف بحيرات البيانات لمعرفة محتوى البحيرة وتنوع طبيعة البيانات التي تغذيها؛ هذا الاستكشاف يمكن ان يتبع بالتعلم الآلي أو استخراج البيانات أو أنشطة أخرى.

الغرض من العمل المعروض في هذه المذكرة هو إنشاء نظام استكشاف حول بحيرات البيانات المكونة من مصادر NoSQL مختلفة، مما يتيح اكتشاف وتصور واستكشاف محتوى بحيرة البيانات بناءً على نموذج البيانات الوصفية الكامل. يتم تطبيق هذا النظام على بيانات شبكة StackExchange من خلال اتباع عملية تطوير TUP2 بطريقة خفيفة، ويمكن توسيعه إلى أنواع أخرى من المصادر.

الكلمات الرئيسية : البيانات الضخمة، بحيرات البيانات، الاستكشاف، NoSQL، البيانات الوصفية، StackExchange، 2TUP.

Résumé

L'exploitation des données massives (Big Data) stockées dans les lacs de données nécessite une étape de découverte et d'exploration des lacs de données pour connaître le contenu du lac, la diversité et nature des données qui l'alimentent ; cette exploration pouvant être suivie par des activités d'apprentissage automatique, de fouille de données ou autres.

Le travail présenté dans ce mémoire a pour objectif de réaliser un système d'exploration autour de lacs de données composé de différentes sources NoSQL, qui permet de découvrir, visualiser et explorer le contenu de lac de données en se basant sur un modèle de métadonnées complet. Ce système est appliqué aux données du réseau StackExchange en suivant le processus de développement 2TUP d'une manière allégée, et peut être extensible à d'autres types de sources.

Mots Clés : Big Data, Lacs de données, Exploration, NoSQL, Métadonnées, StackExchange, 2TUP.

Abstract

The exploitation of massive data (Big Data) stored in data lakes requires a stage of discovery and exploration of data lakes to know the content of the lake, the diversity and nature of the data that feeds it; this exploration can be followed by machine learning, data mining or other activities.

The work presented in this thesis aims to achieve an exploration system around data lakes composed of different NoSQL sources, which allows to discover, visualize and explore the content of data lakes based on a complete metadata model. This system is applied to a dataset from the StackExchange network following the 2TUP development process in a lean way, and can be extended to other types of sources.

Keywords: Big Data, Data Lakes, Exploration, NoSQL, Metadata, StackExchange, 2TUP.

Table de matières

Liste des figures

Liste des tableaux

Introduction générale

Depuis plus de cinq décennies l'informatique s'est implantée au cœur de nos entreprises, nos hôpitaux, nos ministères, nos foyers …Etc. Cette forte utilisation de l'informatique a engendré de grands volumes de données qui ne sont pas gérables par les logiciels et matériels classiques. Prenons le cas d'entreprises géantes comme Google et Microsoft qui gèrent et doivent avoir des volumes jamais imaginés de données à conserver. Un autre exemple est celui des entreprises de téléphonie traitant de grands volumes de données sur les clients et les prestations qui leur sont offertes. Cette perplexité dans la gestion de ces grands volumes de données a donné naissance au Big Data (données massives); qui sont apparus pour traduire à la fois cette explosion de données et la capacité à les traiter en des temps record.

L'expression de *Big Data* est liée principalement à trois caractéristiques majeures : le volume, la variété et la vélocité, qui signifient que ce phénomène ne concentre pas seulement la quantité de données mais fait également référence à leur analyse, leurs types et leur utilisation… En utilisant l'infrastructure de Mégadonnées ou Big Data, les entreprises commencent à rassembler des volumes de données de différents types à des fins d'analyse ou simplement pour les stocker en vue d'une utilisation future indéterminée, le défi ici n'a pas trait seulement au grand volume de données et comment les collecter, mais comment les entreprises être capable de stocker, gérer, analyser et traiter des données de différents types collectées à partir des divers sources avec une haute performance, Pour répondre à ce besoin et relever ces défis, le concept de *Data Lake* ou lac de données a émergé comme un dépôt de stockage qui contient de grandes quantités de données dans leur format natif à utiliser au moment nécessaire, comme pour effectuer l'analyse de données volumineuse qui ingéré des données brutes structurées de manière hétérogène à partir de diverses sources. Les lacs de données fournissant à ce titre un référentiel sans schéma prédéfini pour les données brutes avec une interface d'accès commune.

Aujourd'hui, grâce au lac de données, les utilisateurs pourront matérialiser les données dont ils ont besoin et effectuer des analyses avancées, en plus, il permet aux entreprises de se concentrer sur leur proposition de valeur ajoutée et sur les solutions innovantes qu'elles peuvent mettre en œuvre. Toutefois l'absence du schéma, la vitesse et le volume des données ingérées dans un lac de données peuvent amener les utilisateurs à ne pas connaître le contenu

du lac de données, et donc l'exploration et la découverte du lac de données seront difficiles et impossibles dans certains cas. L'idée donc et que les lacs de données doivent s'appuyer sur des systèmes de gestion de métadonnées efficaces et complets qui aident à dévoiler et découvrir leur contenu et rendre les données interrogeables et accessibles.

L'objectif de ce travail est de fournir la possibilité de découvrir, explorer et visualiser le contenu d'un lac de données en termes de structure, de métadonnées et de données réelles et ce, de manière intégrée, i.e. à travers un seul système.

Notre travail consiste à développer un système qui permet de découvrir et d'explorer un lac de données composé initialement de diverses sources NoSQL (graphes, documents et relationnelles) et guidé par un système de gestion de métadonnées, en se basant sur des données du réseau StackExchange qui est un réseau de sites de questions et réponses traitant de différents sujets variés. Le système est également ouvert et extensible pour prendre en compte d'autres types de sources à travers leur abstraction.

En plus de l'introduction et de la conclusion, notre mémoire est structurée en cinq chapitres, décrits comme suit:

- Dans le chapitre 1 « Concepts de bases », nous développons les principaux concepts concernant les domaines de notre travail : Big Data, Bases de données NoSQL graphes, Base de données NoSQL documents à travers leurs définitions, leurs processus de fonctionnement, et leurs domaines d'application.
- Nous présentons dans le deuxième chapitre « Data Lake » le concept du lac de données, ses origines, son architecture, ses caractéristiques et fonctionnalités … etc., en accordant une importance particulière au mécanisme d'exploration de données dans les lacs de données.
- Dans le chapitre 3 « Capture des besoins fonctionnels », nous présentons en premier lieu le cas d'étude qui sert de fil conducteur tout au long de notre travail. Ensuite nous mettons l'accent sur la capture de besoins fonctionnels du système où nous montrons l'application du processus 2TUP allégé (2-Track Unified Process), en détaillant les deux étapes d'étude préliminaire et de capture des besoins fonctionnels.
- Le chapitre 4 « Analyse et Conception » est consacré à l'analyse et à la conception du système, où nous détaillons en premier lieu l'étape d'analyse de processus 2TUP qui permet d'approfondir l'étape de capture des besoins fonctionnels réalisée dans le

chapitre précédent. Ensuite nous continuons avec la phase de conception de processus 2TUP.

- Enfin dans le chapitre 5 « Mise en œuvre », nous présentons le cadre technique choisi pour développer notre système. Dans un premier lieu, nous présenterons les différents outils et technologies utilisés lors du développement. Ensuite, nous montrons la mise en œuvre de notre application.

Chapitre 01 : Les Big Data

1.1 Introduction

Ces dernières années, nous avons constaté une croissance exponentielle des nouvelles technologies et services qui génèrent un grand volume de données avec divers types, à partir de plusieurs sources internes mais aussi externes et avec des formats divers. Dans ce contexte, le principal besoin exprimé par les utilisateurs et les organisations est de pouvoir gérer ces données afin de les explorer, les exploiter, et les analyser sémantiquement au sein de leurs organisations. Pour répondre à ce besoin, un ensemble de réponse logicielles et matérielles étiquetées « Big Data » a vu le jour. Dans ce chapitre nous allons présenter les principes de base liés aux solutions proposées dans ce contexte : le Big Data, NoSQL avec deux de ses types: les bases de données graphe et les bases de données document.

1.2 Big Data

1.2.1 Origine du Big Data

L'expression « Big data » fait son apparition en octobre 1997 dans la bibliothèque numérique de l'ACM, au sein d'articles scientifiques qui pointent du doigt les défis technologiques à visualiser les « grands ensembles de données », la naissance de ce phénomène est liée à l'explosion quantitative de données numériques; cette explosion ne signifie pas seulement le volume énorme de données mais également sa diversité. Les premiers projets de Big Data sont ceux des acteurs de la recherche d'information sur le web « moteurs de recherche » tels que Google et Yahoo!. En effet, ces acteurs étaient confrontés aux problèmes de la scalabilité (passage à l'échelle) des systèmes et du temps de réponse aux requêtes utilisateurs [1].

Le Big Data devenu une tendance incontournable pour beaucoup d'acteurs industriels du fait de l'apport qu'il offre en qualité de stockage, traitement et d'analyse de données, il consiste à chercher des modèles dans des données à faible densité en informations, à en extraire des faits nouveaux ou de nouvelles relations entre des faits [1] [2].

1.2.2 Qu'est-ce que le Big Data ?

Plusieurs définitions ont été proposées pour décrire le Big Data mais elles sont similaires et renferment les mêmes concepts. Nous retenons la définition suivante:

Selon P. Laflamme[1] : **«** Le Big Data (ou mégadonnées) représente les collections de données caractérisées par un **volume**, une **vélocité** et une **variété** si grands que leur transformation en **valeur** utilisable requiert l'utilisation de technologies et de méthodes analytiques spécifiques. »[3]

1.2.3 Caractéristiques du Big Data

Pour décrire le principe du Big Data, il est nécessaire de résumer ses caractéristiques majeures en utilisant le slogan des 3V (**volume**, **vélocité**, **variété**) :

- **Volume :** le Big Data est généralement à une forte liaison avec cette caractéristique qui décrit la quantité ou la masse de données à collecter, stocker, traiter, et générer par des personnes ou des entreprises. En effet elle désigne la quantité de données qui peuvent être exploitable pour extraire des informations nécessaires pour la prise de décision. Généralement, le volume de données de Big Data ne peut pas être traité par des outils informatiques standards. Pour cela, les entreprises gérant des données massives se voient assujetties à trouver des techniques nécessaires pour gérer les volumes de données collectées chaque jour [4].
- **Vélocité :** représente la fréquence à laquelle les données sont traitées simultanément, cette caractéristique peut faire référence à la disponibilité des données et des résultats en temps réel. Les entreprises doivent appréhender la vitesse non seulement en termes de création de données, mais aussi sur le plan de leur traitement, de leur analyse et de leur restitution à l'utilisateur en respectant les exigences des applications en temps réel [5].
- **Variété :** les technologies Big Data, permettent de créer, intégrer, analyser et classer des données de différents types. La variété désigne l'origine variée des sources de données qui englobe des données structurés (dans les bases de données sur le langage SQL), non structurés (fichiers textes et documents conservés dans des clusters Hadoop ou des systèmes NoSQL), et des données semi-structurées (les journaux des serveurs web ou les données en continu) [6].

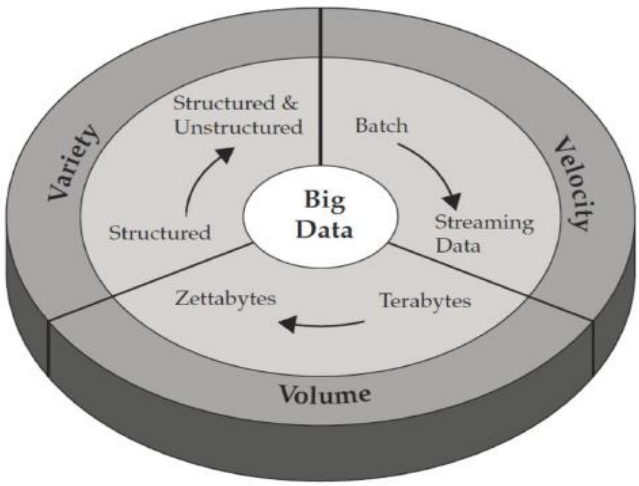


Figure 1.1 : Caractéristiques du Big Data [3].

Le slogan 3V a été complété par d'autres V à l'inspiration des services marketing des acteurs de TI (technologies d'information). Exemples : **Valeur**, **Variabilité**, **Véracité.**

1.2.4 Types du Big Data

Il existe trois types de données dans le Big Data : données structurées, semi-structurées et non-structurées:

- **Données structurées** : ce sont des données qui se basent sur un modèle prédéfini qui se trouvent dans des bases de données relationnelles ; ce type peut être traité automatiquement par des logiciels, et analysé facilement car les données structurées ne souvent pas besoin d'une opération de transformation ou de nettoyage.
- **Données semi-structurées :** les données semi-structurées ou les données partiellement structurées, sont un type de données qui ne se limite pas à une structure rigide, et bien précise telle que celle dans les bases de données relationnelles, mais possèdent des caractéristiques organisationnelles qui facilitent leur analyse telle que les métadonnées ou les balises sémantiques ; le rôle de cette catégorie est de classer les données qui n'appartient pas à l'une ou l'autre catégorie [7].
- **Données non-structurées :** représentent les données qui n'ont pas de structure prédéfinie ou de format spécifique. Elles sont définies comme des données présentes sous forme brute absolu. Ce type de données est difficile à traiter en raison de son formatage complexe, et sa gestion peut prendre un nombre énorme de formes [7].

1.2.5 Techniques et Technologies du Big Data

Les technologies du Big Data sont des techniques et des logiciels permettant la collecte, le traitement, l'analyse et la visualisation de données massives au volume important. Le tableau ci-dessous présente les solutions les plus utilisées dans le Big Data :

Technologie	Description
MapReduce	Framework composé d'un modèle de programmation et sa mise en œuvre. C'est l'un des éléments essentiels de la nouvelle génération de gestion du Big Data-outils de gestion et d'analyse [8].
Hadoop	Framework libre, écrit en java, créé et distribué par la fondation Apache, et destiné au traitement de données volumineuses (de l'ordre des pétaoctets et plus) ainsi qu'à leur gestion intensive [9]. **HDFS** ***(****Le système de fichiers distribué Hadoop*) est un système de fichiers distribué, évolutif et portable écrit en Java, dans laquelle toutes les machines prenant en charge Java peuvent l'exécuter [8].
Bases NoSQL	Les bases de données NoSQL appelées également Cloud Data Bases, sont des outils qui ne sont plus fondés sur l'architecture classique des bases relationnelles. Elles permettent de stocker de grands volumes de données structurées, semi-structurées et non structurées [5].
Stockage « In-Memory »	Une base de données dite « en mémoire » (in-memory), ou IMDB (In Memory DataBase), ou encore MMDB (Main Memory DB), désigne une base de données dont les informations sont stockées en mémoire centrale afin d'accélérer les temps de réponse [10].
Cloud computing	Ensemble de processus permettant d'offrir un espace de stockage sous forme de serveurs, accessibles à distance, sous forme de location, utilisé pour les entités (entreprises) qui ne souhaitent pas investir dans les infrastructures de stockage [11].

Tableau 1.1 : Technologies du Big Data.

1.2.6 Cas d'usage du Big Data

Le Big Data est utilisé dans un grand nombre de secteurs ou domaines d'activités, nous en avons sélectionné quelques-uns pour mettre en avant la grande variété des cas d'usage de Big Data, Des exemples sont cités ci-dessous :

- **Santé** : Aujourd'hui la médecine moderne nécessite l'utilisation d'un grand volume de données personnelles. Des travaux publiés, notamment en cardiologie, dermatologie et diabétologie, montrent la capacité d'expertise de l'exploitation de fichiers par intelligence artificielle avec un niveau aujourd'hui comparable à ceux d'experts de ces spécialités. Les bénéfices potentiels concernent le renforcement de l'expertise et de l'aide au diagnostic pour la pratique médicale et de l'élargissement de la capacité d'analyse pour les chercheurs. Les patients trouveront des améliorations dans la prise en charge personnalisée de leur santé grâce à l'utilisation des objets connectés et en santé publique [12].
- **Marketing :** le développement de l'e-commerce, des réseaux sociaux, du CRM *(Customer Relationship Management*) et de l'IoT *(Internet of Things*) par exemple fait que le domaine de marketing comme les autres a de plus en plus de mal à appréhender des données plus nombreuses, plus complexes, plus variées et qui nécessitent de nouvelles compétences spécifiques pour les assimiler. Pour cela, ce domaine est devenu un client de Big Data, que ce soit pour l'analyse prédictive ou pour l'analyse de sentiments[1], avec pourau but de révolutionner les pratiques en améliorant l'expérience client et la gestion de ses données[13].
- **Transport :** l'analyse de données massives dans le domaine de transport permet un contrôle du trafic en exploitant de données de tous types (GPS, Radars, sondes, etc..) afin de fluidifier le trafic et d'évaluer précisément le temps de transport d'un point à un autre pour optimiser la performance des réseaux de transport [6].

1.3 Bases de données NoSQL

1.3.1 Qu'est-ce qu'une base de données NoSQL

Le terme « **NoSQL** » signifie *Not Only SQL* et désigne un type de stockage de données utilisé dans le Big Data, qui fournit des modèles de bases de données non-relationnelles.

[1]**Analyse de sentiments (Opinion mining)** : est le processus qui consiste à déterminer l'opinion, le jugement qui se cache derrière une série de mots, elle est utilisée par les entreprises et les marques pour comprendre la réponse d'un client à un produit ou un service particulier.

Une base de données NoSQL est une base de données non relationnelle qui se base sur une architecture distribuée caractérisée par une grande performance et haute disponibilité et capable de stocker une large variété de données.

1.3.2 Origine de NoSQL

L'explosion de la volumétrie des données, qui reflète le changement de volume, de nombre et de type de données conduit à l'émergence de l'informatique distribuée qui impose de nouveaux besoins tels que le web scale, les données semi-structurées, etc. Ces besoins ne peuvent pas être atteints avec les systèmes de stockage de données classiques tels que les bases de données relationnelles, et c'est ce qui a forcé les grandes entreprises à chercher de nouveaux outils et techniques plus adaptés à leurs besoins. Dans ce contexte, le NoSQL fut inventé et employé par Carlo Strozzi en 1998 afin d'assouplir les contraintes habituellement présentes sur les bases de données relationnelles. Ce concept a ensuite été adopté et popularisé par les GAFAM (Google, Apple, Facebook, Amazon et Microsoft) qui développent des systèmes NoSQL propriétaires tels que CASSANDRA pour Facebook et BIG TABLE par Google [14].

1.3.3 Type de base de données NoSQL

On distingue actuellement quatre types de bases NoSQL (clé-valeur, orientées-colonne, orientées-document, graphe) [18] :

Type	Description	Exemples
Clé-valeur	Chaque donnée est identifiée par une clé et une valeur quelconque.	Memcached
Orientée colonne	Les données sont sauvegardées dans des colonnes.	Cassandra
Orientée-document	Spécialisation du concept clé-valeur où les valeurs sont des documents d'un certain format (XML, JSON, BSON, etc.)	CouchDB, MongoDB
Graphe	Basée sur la théorie de graphe, telle que les nœuds représentent les objets et les liens entre les nœuds représentent les relations entre les objets.	Neo4j

Tableau1.2 : Types de base de données NoSQL.

Chacune de ces catégories présente des caractéristiques différentes en termes de scalabilité horizontale. Par exemple, les bases orientées graphes ne passent pas aussi facilement à l'échelle horizontalement, mais sont pourtant indispensables pour traiter efficacement les données issues des réseaux sociaux. Dans notre travail, nous sommes particulièrement intéressées par les bases de données graphes et les bases de données orientées document, que nous détaillons ci-dessous.

1.4 Base de données orientées graphe

L'apparition du NoSQL avec ses quatre types de bases de données (clé-valeur, colonne, document, graphe) apporte une réponse aux problèmes posés par la base de données relationnelle classique, qui atteint rapidement ses limites dans le cas de masses de données très volumineuses et complexes. Bien que les bases de données de type clé-valeur, colonne, ou document tirent leur principal avantage de la performance du traitement de données, les bases de données orientées graphe permettent de résoudre des problèmes très complexes que les bases de données relationnelles seraient incapables de faire. Les réseaux sociaux (Facebook, Twitter, etc.) où des millions d'utilisateurs sont reliés de différentes manières, constituent un bon exemple : amis, fans, famille etc. Le défi ici n'est pas le nombre d'éléments à gérer, mais le nombre de relations qu'il peut y avoir entre tous ces éléments. En effet, il y a potentiellement n^2 relations à stocker pour *n* éléments. Même s'il existe des solutions comme les jointures, les bases de données relationnelles se confrontent très vite à des problèmes de performances ainsi que des problèmes de complexité dans l'élaboration des requêtes. L'approche par graphes devient donc inévitable pour les réseaux sociaux.

1.4.1 Qu'est-ce qu'une base de données orientée graphe (Graph Database) ?

Une base de données orientée graphe est une base de données non-relationnelle ne possédant pas de schéma de données. Comme son nom l'indique est une représentation basée sur des graphes et conçue pour stocker et rechercher des informations qui sont connectées les unes aux autres de manière complexe, ainsi que leurs relations les unes avec les autres [19].

1.4.2 Structure d'une base de données orientée graphe

Une base de données orientée graphe est composée essentiellement de nœuds (*Nodes)*, et d'arcs *(Edges)* [19] :

– Chaque nœud représente une entité (comme une personne ou une entreprise), et est défini par :

- Un identifiant unique.
- Une liste d'arcs sortants.
- Une liste d'arc entrants.

- Une liste de propriétés sous forme de couple clé-valeurs.

Un ensemble de nœuds peuvent être regroupés par des labels.

– Chaque arc représente une connexion entre deux nœuds, et est défini par :

- Un identifiant unique.
- Un nœud de départ.
- Un nœud d'arrivée.
- Un ensemble de propriétés.

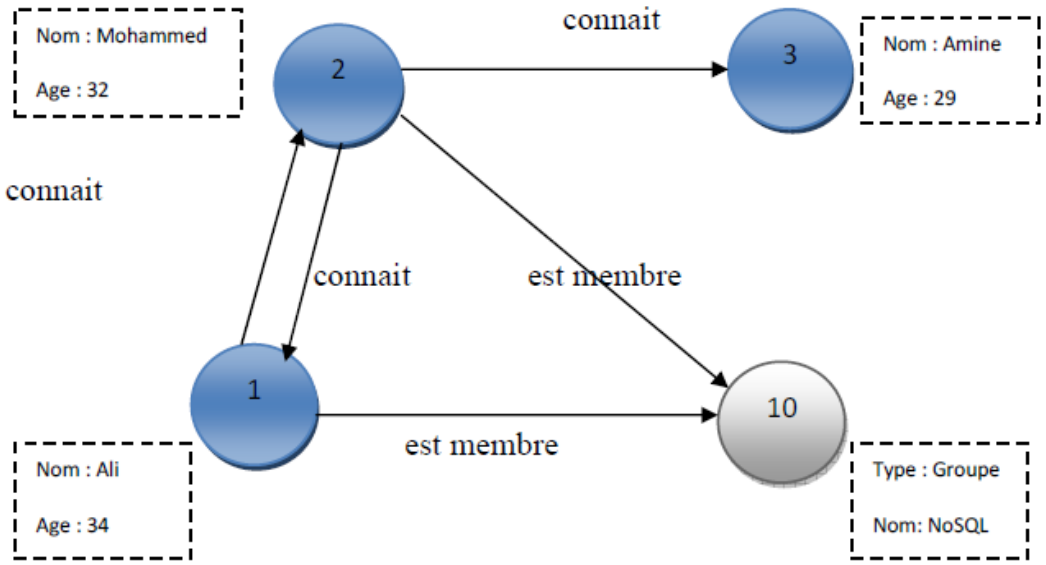


Figure 1.2 : Illustration d'une Base de données Orientée Graphe [20].

1.4.3 Caractéristiques d'une base de données orientée graphe

Pour mieux comprendre la structure et le fonctionnement des bases de données orientées graphes, il est important de savoir quelles sont les caractéristiques qui distinguent ce type de base de données des autres types. Pour cela, nous mettrons en évidence les bases de données orientées graphe par rapport à trois composants de base, à savoir la structure de données, le langage de transformation et les contraintes d'intégrités.

Un modèle de base de données orientée graphe est caractérisée par ce qui suit [21] :

- Les données et le schéma sont représentés par des graphes ou par des structures de données généralisant la notion de graphe (hyper graphes ou hyper nœuds) et basant sur la théorie de graphe.
- La manipulation des données est exprimée par des transformations de graphes ou par des opérations qui se basent sur les caractéristiques de graphe telles que : les chemins, les

voisinages, les sous-graphes, les motifs de graphe, la connectivité et les statistiques de graphe.

- Les contraintes d'intégrité : c'est un concept très important dans la manipulation et le stockage de données qui renforce la cohérence de données. Dans ce type de bases de données, ces contraints peuvent être regroupées en trois éléments : cohérence schéma-instance, identité ou intégrité référentielle et dépendance fonctionnelle. Par exemple, des labels (étiquettes) qui regroupent les nœuds avec des noms uniques (identité), contraintes de typage sur les nœuds, domaine et gamme de propriétés.

  En résumé un modèle de base de données orientée graphe est un modèle dans lequel la structure est modélisée sous forme d'un graphe avec des étiquettes, et où la manipulation de données est exprimée par des opérations autour des graphes, et les contraints d'intégrité sont définies sur la structure du graphe.

1.4.4 Types de bases de données orientées graphes

Il existe deux modèles de base de données graphes courants : les graphes RDF et les graphes de propriétés ; ils ont des points similaires, mais sont construit en mettant l'accent sur des défis différents. Le RDF se concentre sur l'intégration des données, tandis que les graphes de propriétés se concentrent sur l'analyse de données.

- **RDF (Ressource Description Framework)**: n'est pas à proprement parler un langage, il s'agit plutôt d'un modèle de données pour décrire des ressources sur le Web, mais pas nécessairement accessibles sur le web.

  En RDF les données sont liées via une structure « sujet-prédicat-objet », tandis que le sujet est un nœud, le prédicat est une arête, et l'objet est soit un autre nœud, soit un littéral. Ce triplet RDF peut être modélisé sous la forme d'un graphe nommé par un identifiant de ressource unique URI (*Uniform Resource Identifier*) [22].

- **Graphe de propriété** : Un modèle de graphe populaire à côté de RDF est le graphe de propriété qui est considéré comme un multi-graphe orienté, dirigé et étiqueté [23].

  Les LPG (*LabelledProperty Graph*) peuvent supporter des propriétés directement dans leurs nœuds et leurs relations ; la structure interne de ces nœuds et relations est décrite par des paires Clé-Valeur. Ils sont généralement centrés et orientés sur les nœuds, différents des triplets RDF centrés sur les bords [22].

1.4.5 Langage de requête et de manipulation

Un langage de requêtes est un ensemble d'opérateurs ou de règles d'inférence, utilisés pour accéder, manipuler, et interroger les données d'une base de données. Dans le contexte des modèles de données orientés graphe, il existe des langages de requête qui prennent en compte

la transformation de base de données en tant que transformation de graphes, ils sont basés sur l'appariement de motifs de graphes et permettent à l'utilisateur d'accéder et spécifier les insertions et les suppressions des nœuds de manière graphique [21].

Les bases de données orienté graphe proposent différents langages de requête tels que :

SPARQL : est l'acronyme de *Simple Protocol And RDF QueryLanguage,* est un langage de requête standard développé par W3C en 2008, adapté à la structure spécifique des graphes RDF. Il joue le rôle d'un pont entre les technologies du Web sémantique (dont RDF), et les plateformes Web déjà existantes. Il est une API universelle d'accès aux données [24].

Cypher : c'est un langage d'interrogation déclaratif inspiré par SQL et SPARQL, inventé par NEO4J dans le but de la mise en main, la lecture et l'écriture des requêtes, il permet une interrogation et une mise à jour efficaces du Triples Store [24].

GraphQL : créé par Facebook, c'est un langage de requête pour les API qui n'est pas spécifique aux bases de données de graphes. Les utilisateurs définissent la structure des données dont ils ont besoin et obtiennent exactement ce qu'ils ont demandé. Les requêtes GraphQL sont organisées en types et en champs, et non en points de terminaison. En modifiant l'objet de requête, vous pouvez déterminer ce qui est renvoyé par le serveur [4].

Gremlin: il a été créé en 2009 et est le langage de requête pour Apache TinkerPop. C'est un DSL (Domain SpecificLanguage) de parcours de graphe qui peut être déclaratif ou impératif. Il peut être utilisé pour les bases de données orientées graphes OLTP et OLAP. Gremlin est basé sur Groovy, mais possède de nombreuses variantes permettant aux développeurs d'écrire des requêtes Gremlin de manière native dans de nombreux langages de programmation modernes tels que Java, JavaScript, Python, Scala, Clojure et Groovy[4].

1.4.6 Cas d'utilisation des bases de données orientées graphes [27]

Les bases de données orientées graphes peuvent être appliquées pour résoudre n'importe quel problème dans tout domaine qui est très riche en liens entre les entités, c'est-à-dire les domaines qui nécessitent de collecter et relier des points de données.

Voici les principaux cas d'utilisation des technologies de base de données orientées graphes :

- **Détection de fraudes:** Les bases de données orientées graphe peuvent détecter les fraudes de manière sophistiquées, en raison qu'elles relèvent des modèles difficiles à détecter à l'aide de représentations traditionnelle telles que les tableaux, avec ce type de

base de données les entreprises et les organisations peuvent utiliser les relations pour traiter les différents transactions, et détecter tous types des fraudes qui les suivent, en formulent des requêtes de graphe rapides surtout dans le cas des processus commerciaux qui deviennent plus rapides et plus automatisés.

- **Moteur de recommandation :** les bases de données orientées graphe sont un bon choix pour les applications de recommandation, et aussi la technologie-clé qui permet des recommandations en temps réel. Faire une recommandation efficace dépend d'une base de données qui comprend les relations entre les entités, ainsi que la qualité et la force de connexion, seule une base de données orientée graphe suit efficacement ces relations en fonction des achats, des interactions et des avis des utilisateurs pour donner l'aperçu le plus significatif des besoins des clients et tendances des produits.
- **Gestion des réseaux informatiques** : De par la nature même, les réseaux sont des graphes, les bases de données orientées graphe sont donc un excellent choix pour modéliser, stocker et interroger les données opérationnelles du réseau et de l'ordinateur. Aujourd'hui, les bases de données graphes travaillent avec succès dans les domaines des télécommunications, de la gestion des réseaux, de l'analyse d'impact, de la gestion des plateformes Cloud, et de la gestion des centres de données, et des actifs informatiques.

1.4.7 Avantages et limites des bases de données orientées graphe

1.4.7.1 Avantages

Le point positif de ce type de base de données est qu'elles sont parfaitement adaptées à la gestion de données relationnelles, même dans un contexte du Big Data. De plus, les modèles orientés graphes permettant beaucoup de souplesse dans la représentation des données à l'aide des nœuds et des arcs ; pour cela leur architecture étant modulable, elle peut être adaptée selon les besoins rencontrés [25].

1.4.7.2 Limites

Les bases de données de graphes partagent le point commun des bases de données NoSQL: l'absence de langage de requête uniforme. Bien que cela puisse être un obstacle à l'utilisation d'une base de données, cela n'affecte pas les performances de ce type de base de données. Un autre inconvénient est l'évolutivité de ces bases de données, car elles sont conçues pour une architecture à un niveau, ce qui signifie qu'elles sont difficiles à faire évoluer sur plusieurs serveurs. Comme toutes les autres bases de données NoSQL, elles sont conçues pour servir un objectif spécifique et y exceller. Il ne s'agit pas d'une solution universelle conçue pour remplacer toutes les autres bases de données [44].

1.5 Base de données orientée document

Selon le dictionnaire d'Oxford en ligne, un document est un morceau de matériel écrit, imprimé ou électronique qui fournit des informations ou des preuves ou qui sert de dossier officiel [28]. Dans ce contexte, il existe des bases de données basées sur des documents pour stocker des informations, tandis que la structure de ces documents dépend de l'implémentation et la nature de données.

1.5.1 Qu'est-ce qu'une base de données orientée document (document Database) ?

Une base de données orientée document est un type de base de données non relationnelle, conçue pour stocker et interroger des données sous forme de documents de type : JSON, XML. Ces bases de données sont une évolution des bases de données de type « clé-valeur » mais avec le principe « clé –document », bien que les documents soient structurés, ce type de base est appelé « *Schemaless* », il n'est donc pas nécessaire de définir les champs d'un document au préalable [25].

1.5.2 Structure d'une base de données orientée document

- Un modèle ou une base de données orientée document est organisé en collection de documents.
- Un document est construit d'un identifiant clé et d'une valeur composée de couple clé-valeur qui peut être hiérarchisé, cela veut dire qu'une valeur peut elle-même contenir une ou plusieurs paires clé-valeur.
- Dans la même collection le document peut être défini par des structures différentes, d'autre terme avec des couples qui n'ont pas nécessairement les mêmes noms.

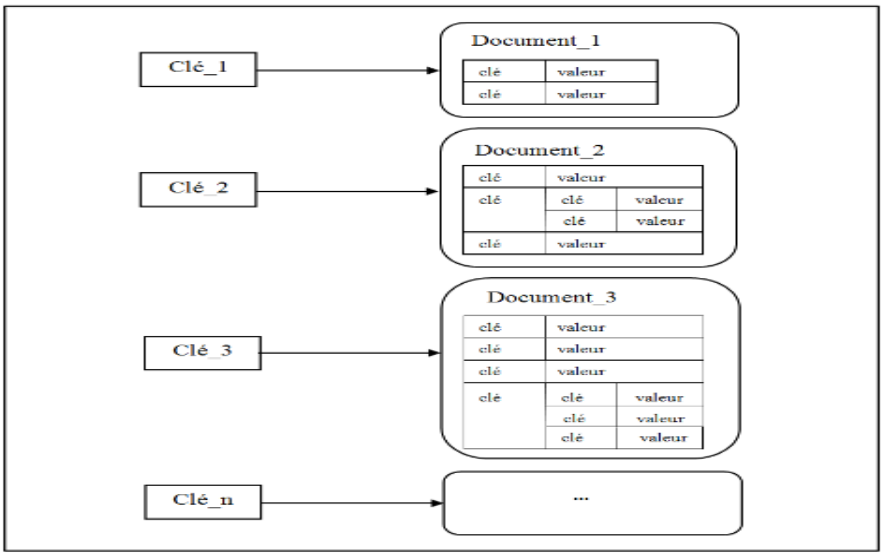


Figure 1.3 : Organisation d'une collection dans une BD orientée-documents [29].

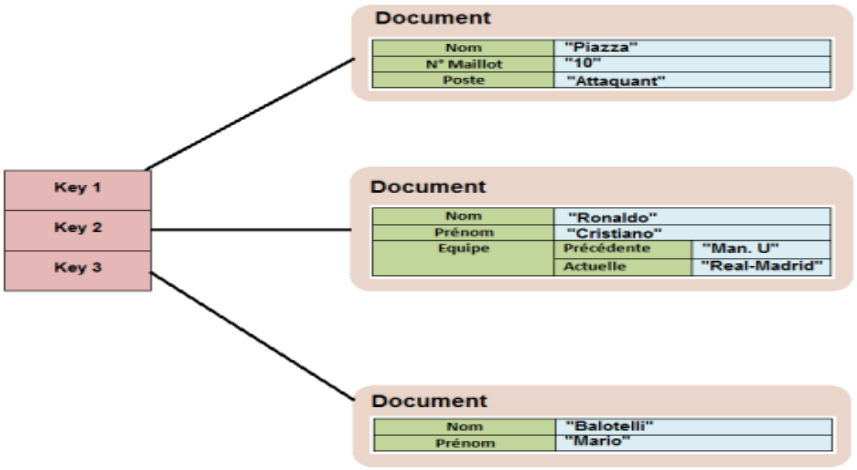


Figure 1.4 : Illustration d'une Base de données Orientée document [25].

1.5.3 Caractéristiques des bases de données orientées document

Un modèle de base de données orientée document peut être caractérisé par ce qui suit :

- **Schemaless** : il n'y a aucune restriction au format et structure de stockage de données ; dans ce type de base de données il est possible de stocker des documents avec un ensemble d'attributs complètement différents [30].
- **Indépendance** : les documents sont indépendants les uns des autres dans les SGBD document, il n'est pas possible de modéliser des liens ou des documents reliés entre eux.
- **Mapping direct avec les langages de programmation** : les bases de données orientées documents permettent aux développeurs de stocker et interroger une base de données en utilisant le même format de modèle de document que celui qu'ils utilisent dans leur code d'application, en raison que les documents peuvent être mappés directement à la structure du langage de programmation [30].
- **Structure arborescente :** un modèle orienté document supporte la notion d'imbrication ou la structure arborescente ; cela veut dire que le couple clé-valeur qui considéré comme un identifiant de document peut être hiérarchisé (une valeur peut-elle-même contenir une ou plusieurs pair clé-valeur).

1.5.4 Fonctionnement des bases de données orientées document [31]

Dans la théorie, une base de données orientée document peut contenir des données avec des formats différents ; cependant dans le fonctionnement du ce type de base de données, on utilise un format de fichier et l'on intègre les informations dans une structure fixe, les bases de données document récupèrent et reçoivent des données sous forme des paires clé-valeur,

mais ici les valeurs sont des documents , tandis que chaque document possède un clé unique pour l'identifier, afin d'accéder aux données de la base de données, ces identifiants ont tendance à être indexés dans les bases de données, pour accélérer la récupération des informations.

Le contenu des documents dans ce type de base de données est classé à l'aide de métadonnées. Grâce à cette fonctionnalité, la base de données "comprend" quelle catégorie d'informations elle contient. Pour une efficacité et une expérience utilisateur améliorée, les magasins de documents disposent d'un langage de requête, qui permet d'interroger des documents en fonction des métadonnées ou du contenu du document. Cela vous permet de récupérer tous les documents qui contiennent la même valeur dans un certain champ.

1.5.5 Vue d'ensemble de bases de données orientées document les plus populaires

Nous proposons dans ce tableau les deux bases de données les plus utilisées [20] :

	MongoDB	**CouchBase**
Présentation	SGBD orienté document développé depuis 2007 par la société de logiciel 10gen	SGBD orienté document conçu pour les applications web interactives.
Distribution	-Licence AGPL	-Apache http server
Architecture	- il est développé en C++ -Il possède différents modes de fonctionnement tels que single, Réplication, Sharding	-il utilise un cluster couchBase pour connecter les applications, et les SDKs pour les écrits dans les langages de programmation.

Tableau 1.3 : Les bases de données orientées document.

1.5.6 Cas d'utilisation [32]

Les bases de données de documents sont utiles pour les charges de travail qui nécessitent un schéma flexible pour un développement rapide et itératif. Voici quelques exemples de cas d'utilisation pour les bases de données orientés document

- **Persistance de profils utilisateurs :** Les bases de données documentaires sont une solution pratique aux profils en ligne dans lesquels différents utilisateurs fournissent différents types d'informations. Étant donné que ce type de base de données possède un schéma flexible, ils peuvent stocker des documents qui ont des attributs et des valeurs de données différents.
- **Big Data en temps réel :** Pouvoir extraire des informations opérationnelles en temps réel est essentiel dans un environnement commercial hautement concurrentiel. En

utilisant des bases de données de documents, une entreprise peut stocker et gérer des données opérationnelles à partir de n'importe quelle source et transmettre simultanément les données au moteur de BI de son choix pour analyse.

●**Gestion de contenu :** une base de données orientée document est un bon choix pour les applications de gestion de contenu comme les blogs et les plateformes vidéo. Avec ce type de base de données, chaque entité qui suit l'application peut être stockée comme un document unique, où il est possible de créer et incorporer de nouveaux contenus, y compris du contenu généré par l'utilisateur, comme des images, des commentaires et des vidéos.

1.5.7 Avantages et limites des bases de données orientées document [25]

L'avantage de base de données orientée document est de pouvoir récupérer un ensemble d'informations structurées hiérarchiquement depuis une clé. Cette opération est similaire à la jointure qui peut être coûteuse dans le modèle relationnel. La nature souple, semi structurée et hiérarchique des documents et des bases de données de document leurs permet d'évoluer avec les besoins d'applications; ce type de stockage est par conséquent très convoité par les applications web diffusant des pages entières obtenus par un ensemble des jointures.

La duplication des informations est un point négatif très connu dans les bases de données orientées document, les données étant regroupées par documents, il se pourrait que des problèmes d'intégrités surgissent, certaines données étant nécessairement dupliquées sur plusieurs documents, cette duplication peut affecter la cohérence des données. Dans le cas de volume de données fortement liées, l'utilisation des documents rend le système très complexe et peu pratique pour cela ce type de base de données est inadapté aux domaines riches en liens.

1.6 Conclusion

Dans ce chapitre, nous avons présenté d'une façon détaillée le phénomène permettant d'étudier de très grandes quantités de données qui s'appelle « Big Data », qui peut être considérée comme un écosystème très large et complexe, qui dépasse la capacité des systèmes traditionnels en termes de stockage et de traitement de données. Notre présentation de cette approche inclut son origine, ses caractéristiques, ses technologies, ses cas d'usage et également son type de stockage de données «NoSQL» qui fournit des modèles de base de données non relationnelles. Dans ce contexte nous avons détaillé deux types de bases de données NOSQL : les bases de données orientées document et les bases de données orientées graphe.

Chapitre 02 : Les lacs de données (Data Lakes)

2.1 Introduction

Le Big Data est sans aucun doute devenu l'un des plus importants défis dans la recherche sur les bases de données. Un volume énorme, une grande variété et une vitesse élevée de données doivent être collectées, stockées et traitées avec une qualité de données élevée (véracité) pour fournir de la valeur. Ces évolutions constantes créent de plus en plus de besoins en technologies performantes d'analyse des données qui dépassent les outils classiques. Le principal inconvénient de ces outils est la difficulté pour les utilisateurs d'explorer de manière flexible un ensemble de données massives et hétérogènes et stockées sur divers supports de stockage. Pour combler cette lacune et répondre à ce besoin, les lacs de données ont été proposés comme dépôts de mégadonnées, qui stockent des données brutes et fournissent une liste riche de fonctionnalités avec l'aide des descriptions des métadonnées. Dans ce chapitre nous allons présenter le concept de lac de données, ses origines, son architecture, ses caractéristiques et fonctionnalités … etc., en accordant une importance particulière au mécanisme d'exploration de données dans les lacs de données.

2.2 Data Lake

2.2.1 Qu'est-ce qu'un Data Lake

Le concept de Data Lake (ou Lac de données[2]) a été introduit par Dixon en 2010 comme une alternative aux magasins de données (Data Mart). Dixon définit les lacs de données en tant qu'un vaste dépôt de données brutes de structures hétérogènes alimentées par des sources de données externes [33].

Après Dixon, de nombreux chercheurs ont proposé des définitions de lac de données, Madera et Laurent [34] considèrent les lacs de données comme une nouvelle solution d'architecture de données composée à la fois de matériel, de logiciel, et de désigne conceptuel, et donc non limitée à une méthodologie. Ravat et Zhao [35] ont identifié les lacs de données en tant qu'une solution d'analyse de données volumineuse qui ingère des données brutes structurées de manière hétérogène à partir de diverses sources. [34]

[2] Nous utiliserons les termes lacs de données ou data Lake de manière inter-changeable.

2.2.2. Origine du Data Lake. L'évolution des systèmes d'aide à la décision sous l'influence de la gouvernance des données

La mission des systèmes d'aide à la décision a toujours été de collecter un ensemble de sources de données pertinentes, les organiser, les structurer afin d'extraire des nouvelles données qui peuvent être exploitées par les décideurs, dans le processus d'aide à la décision. Les informations requises sont connues à l'avance et elles sont de ce fait structurées et optimisées, notamment des entrepôts de données. Avec la naissance de concept de *Big Data,* les utilisateurs des systèmes d'aide à la décision sont confrontés à de nouveaux défis, et ils demandent des solutions pour trouver des idées afin d'exploiter toutes les données disponibles dans les sources, quel que soit la nature et le type de ses informations sans avoir au préalable les besoins d'exploitation. Les utilisateurs ont donc besoin d'un environnement incubateur qui rassemble un grand volume de données de différents types, avec une exploitation de type ELT (extraction, chargement et transformation). C'est de cet environnement incubateur qu'est né le concept de lacs de données.

2.2.3 Architecture fonctionnelle de Data Lake

L'architecture fonctionnelle du lac de données a évolué de mono-zone à multizone, elle est toujours présentée avec des solutions techniques [35] :

Architecture mono-zone : architecture plate composée d'une zone unique, qui sert à stocker des données brutes dans leur format natif; cette architecture est fortement liée à l'environnement Hadoop.

Architecture multizone : cette architecture est composée de cinq bassins de données (bassin de données bruts, de données analogiques, de données textuelles, de données d'applications, et de données d'archives).

De nombreuses architectures fonctionnelles du lacs de données ont été proposées dans les domaines académiques, Ravat et Zhao [35] ont proposé une architecture fonctionnelle du lac de données en la distinguant de l'architecture technique, l'architecture fonctionnelle proposée par ces deux chercheurs contient quatre zone essentielles (voir Figure 2.1) telles que chaque zone à l'exception de la zone régie (Govern Zone), a une zone de traitement (Rectangle en pointillés), et une zone de stockage (rectangle gris).

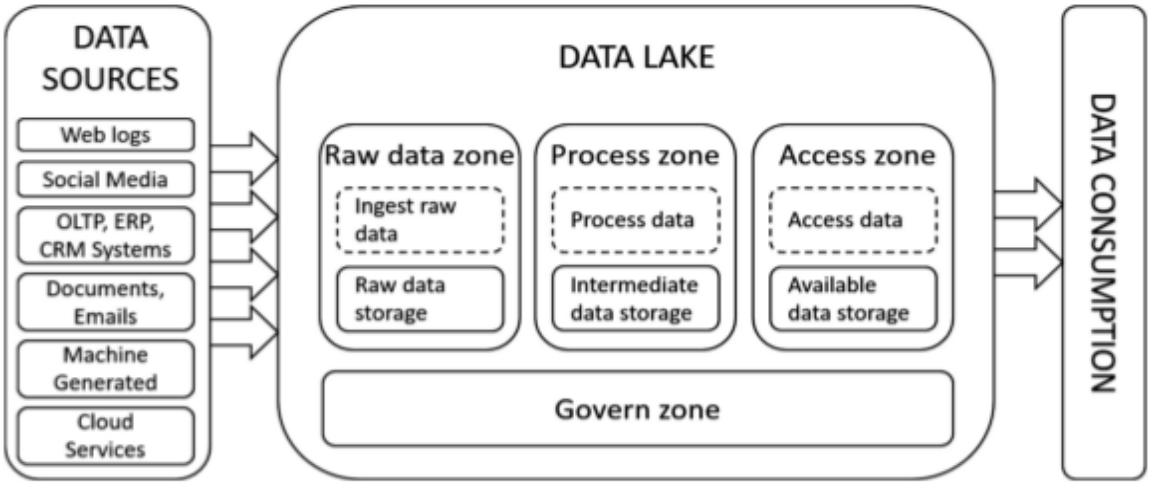


Figure 2.1 : Architecture fonctionnelle du lac de données [35].

La description des zones du (Figure1.9) est présentée dans le tableau suivant [35].

Zone	Description
Raw Data Zone (zone de donnés bruts)	Stocke des données dans leurs formats natifs ; elle permet aux utilisateurs de trouver les versions originales de données pour leurs analyses.
Process zone (zone de traitement)	Permet aux utilisateurs de traiter les données (sélection, projection, jointure, agrégation.) pour leur analyse.
Access zone (zone d'accès)	Stocke toutes les données disponibles et prêtes pour les opérations d'analyse ; elle permet aux utilisateurs d'accéder et consommer ces données pour différentes analyses.
Govern zone (zone de gouvernance)	S'applique à toutes les autres zones ; elle assure la sécurité, la qualité et l'accès de données.

Tableau 2.1 : Description des zones d'architecture fonctionnelle de Data Lake.

2.2.4 A quoi sert un Data Lake ?

Un lac de données fournit des fonctionnalités-clés qui permettent à une entreprise d'améliorer ses processus d'aide à la décision. Le concept de lac de données peut aider à résoudre le problème persistant de l'accessibilité et de l'intégration des données et toutes les difficultés de gestion de données surtout pour les entreprises qui utilisaient le Big Data.

Les approches classiques de l'intégration des données à grand échelle ont forcé tous les utilisateurs d'utiliser un schéma commun avec une structure fixe prédéterminée,

contrairement au lac de données qui autorise et assouplit la normalisation et diffère la modélisation, et de ce fait, fournit une grande possibilité de découvrir plus de données, et cela conduit à une richesse d'exploitation des données et métadonnées et augmente sa valeur ajoutée [36].

2.2.5 Fonctionnalités et caractéristiques des lacs de données

Un lac de données est une structure de stockage, ayant les caractéristiques et les fonctionnalités suivantes :

2.2.5.1 Caractéristiques [37]

- Un catalogue de métadonnées qui facilité l'accès aux données et en assure la qualité.
- Des outils de gestion de données.
- L'accessibilité aux utilisateurs.
- L'évolution possible des données de toute nature.
- Une organisation logique et physique des données.

2.2.5.2 Fonctionnalités [38]

- **Acquisition de données :** Un lac de données permet d'intégrer des sources de données internes, externes, structurées et non structurées.
- **Catalogage de données :** Cette fonctionnalité est reliée au terme « métadonnées », un lac de données repose sur un catalogue de métadonnées permettant de l'utiliser, entre autres, à des fins d'analyse.
- **Stockage de données :** Après l'acquisition et le catalogage de données, ces dernières peuvent être stockées avant, pendant, et après la phase d'exploitation.
- **Exploitation ou exploration de données :** Cette fonctionnalité peut être considérée comme l'objectif des lacs de données, où les données peuvent être exploitées par l'organisation dans le processus de décision par exemple.
- **Gouvernance :** la gouvernance du lac de données vise à gérer le catalogue des métas donnés, la sécurité et le cycle de vie et sa qualité.

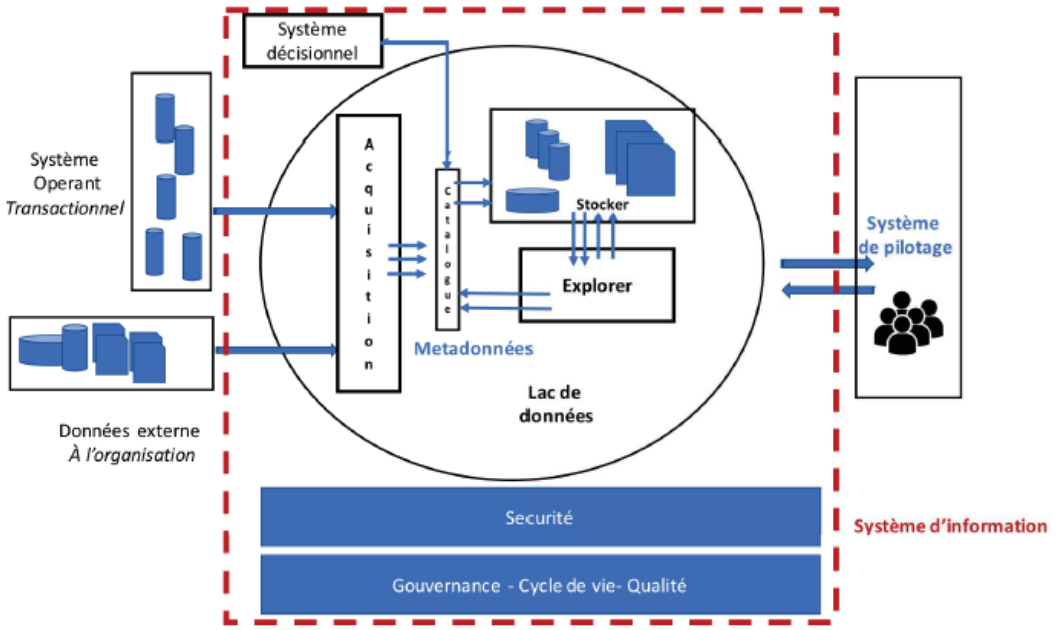


Figure 2.2 : Macro composant ou fonction d'un lac de données [38].

2.2.6 Intégration des lacs de données dans le système d'information

Dans un lac de données, différents utilisateurs peuvent accéder aux données et les traiter à des fins d'exploration de données, ou d'analyse statistique à des fins de prise de décision [35]. Les projets des lacs de données sont, de ce fait, associés à des projets de type **« Analytique »**, tels que les projets décisionnels dans les entreprises [38]. Les points de vue de l'emplacement des lacs de données dans le système d'informations diffèrent d'une étude à une autre : certains auteurs pensent que le meilleur emplacement de lac de données est dans le processus décisionnel à côté de l'entrepôt de données, tandis que d'autres pensent que les lacs de données et les systèmes décisionnels doivent être séparés avec des liens indirects entre eux. Dans [38], Madera considère le lac de données en tant que système dirigé par les données (Data Driven); il a donc une fonction différente du système décisionnel, qui lui est dirigé par information, il a de plus comme objectif d'aider au système de pilotage d'une organisation, et donc sa place dans les SI et au côté des systèmes décisionnel. (Voir **Figure 2.3**).

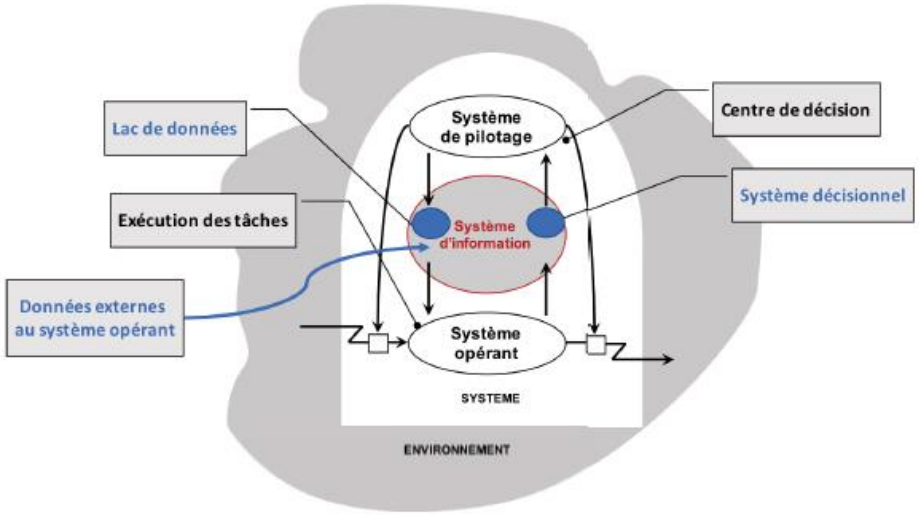


Figure 2.3 : Proposition de positionnement des Lacs de données dans le Système d'information [38].

2.2.7 Data Lake et métadonnées

Une métadonnée est une donnée servant à définir ou décrire une autre donnée quel que soit son support. D'autre part, l'idée principale du lac de données est d'ingérer des données brutes dans leur format natif ; il prône un nouveau paradigme de stockage et d'analyse à travers une approche *Schéma-on-Read*, qui rend l'exploitation des données stockées dans les lacs de données difficile et impossible dans certains cas, en raison que les données sont stockées sans des modèles ou des descriptions explicites. L'idée donc est de placer un système de gestion des métadonnées dans les lacs de données afin de rendre les données interrogeables et accessibles [33] [35].

Les fonctionnalités attendues du système de gestion de métadonnées dans un lac de données sont : [33]

- **Enrichissement sémantique (ES) :** consiste à générer une description du contexte des données.
- **Indexation des données (ID) :** mécanisme de recherche avancée qui consiste à mettre en place une structure de données pour trouver les données sur la base de caractéristiques précis.
- **Génération et conservation des liaisons (GL) :** consiste à détecter les relations de similarité et intégrer les liaisons entre les données.
- **Polymorphisme des données (PD) :** consiste à affecter plusieurs représentations à la même donnée.
- **Versionnement de données (VD) :** consiste à gérer les évolutions de données en conservant leurs états précédents.

- **Suivi d'utilisation (SU) :** consiste à tracer les interactions entre les utilisateurs du lac et les données.

2.2.8 Technologies du data Lake

La plupart des implémentations de lacs de données sont basées sur le Framework **Apache Hadoop** ; cet écosystème a comme avantage de fournir à la fois le stockage de données avec le **HDFS** (*Hadoop Distributed Système*) et le traitement des données avec MapReduce ou Spark. En effet, il existe d'autres solutions techniques appropriées pour implémenter un lac de données, nous allons présenter ci-dessus pour chaque fonctionnalité, l'ensemble d'outils utilisables pour la mettre en œuvre [39] :

- **Ingestion de données :** signifie le transfert physique de données à partir des sources des données vers un lac de données, la plupart des outils pour mettre en œuvre cette phase sont proposée par la fondation Apache, ils incluent : FLINK et SAMZA (Framework de flux distribués), FLUME (Service pour transférer les journaux Hadoop), KAFKA (un Framework fournissant des pipelines de données en temps réel), et SQOOP (Framework d'intégration de données).
- **Stockage de données :** pour le stockage de données dans des lacs de données, on trouve les SGBD relationnels tels que MySQL, postgres SQL, Oracle, les SGBD NoSQL, et le système de stockage distribué le HDFS qui est le plus utilisé.
- **Traitement de données :** Dans les lacs de données, le traitement de données est très souvent réalisé avec « MapReduce » en raison qu'il est le plus adapté aux données volumineuses, il existe d'autres outils de traitement, dont le plus connus est Apache SPARK qui a un fonctionnement un peu similaire à celui de MapReduce.
- **Accès aux données :** Dans les lacs de données, les données peuvent être accessibles avec les langages de requêtes classiques tels que SQL, JSONIq, XQuery, SparQL. Cependant, ces SGBD classiques ne permettent pas un accès simultané aux bases de données hétérogènes ; pour cela de nouveaux outils sont proposés pour résoudre ce problème tel que Spark SQL, SQL^{++}, SQRE, et Cloud Md SQL.

2.3 Exploration des données et lacs de données

2.3.1 Définition de l'exploration de données

L'exploration de données ou fouille de données ou data mining en anglais est un domaine en pleine croissance de l'écosystème de données complexes avec divers types de données collectées via de multiples sources [40]. Cette technique permet d'extraire des informations et des connaissances importantes à partir d'un vaste ensemble de données.

2.3.2 Processus d'exploration de données [41]

Il est très important de comprendre que l'exploration de données n'est pas seulement le problème de découverte de modèles dans un ensemble de données. C'est un processus suivi par les scientifiques et les ingénieurs ou toute autre personne qui cherche à extraire les connaissances à partir des données. L'exploration des données est un processus composé de 4 étapes (Figure 2.4) dont chacune est essentielle pour la suivante. Ceci va du démarrage par les données brutes jusqu'à l'obtention des connaissances comme résultat final.

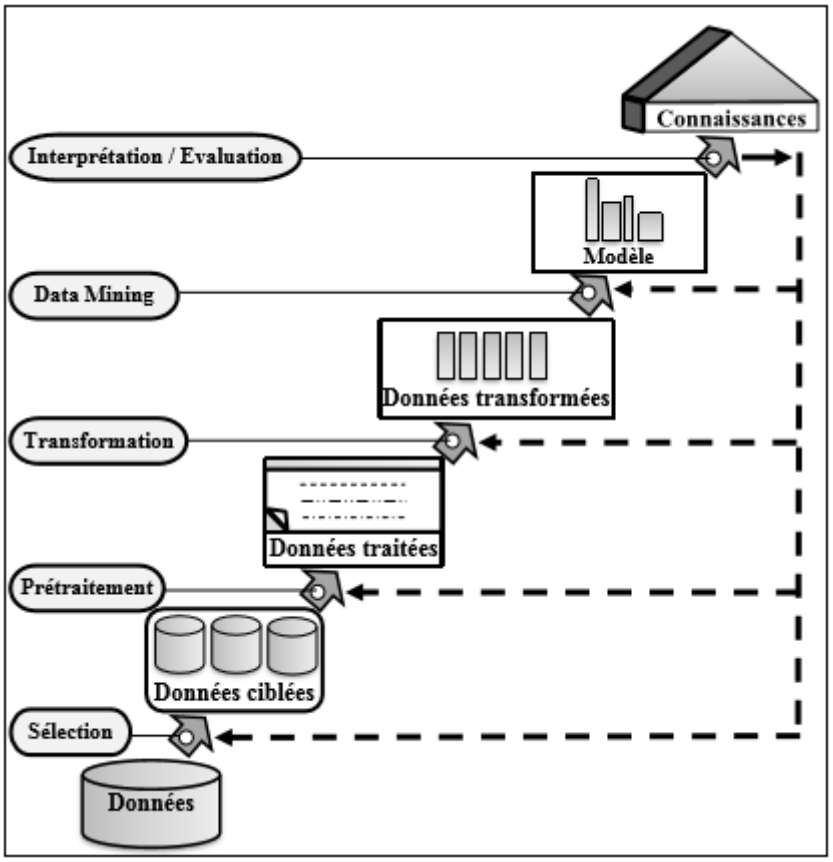


Figure 2.4 : Étapes de processus d'exploration de données [41].

1. **Sélection :** la première étape d'exploration de données consiste à sélectionner les sources de données qui peuvent être utilisées pour extraire et obtenir des informations précieuses.
2. **Transformation de données :** C'est l'opération de codage des données, le processus passe par plusieurs phrases lors de cette étape, la conversion ou changement des types, la normalisation des données, et la rédaction de données.
3. **Data mining (Exploration de données) :** elle implique le choix d'une technique ou d'un algorithme précis pour les pratiquer sur une portion de l'ensemble de données de l'étape précédente.

4. **Évaluation :** consiste à évaluer les résultats de data mining, l'exploration de ces résultats est en plein cœur de cette étape.

2.3.3 Principales tâches d'Exploration de données [42]

Le tableau suivant indique les tâches les plus courantes que l'exploration de données est amenée à accomplir qui sont classification, Estimation, Le groupement par similitude, l'analyse des clusters et la description.

Tâche	Description
Classification	C'est une tâche supervisée qui consiste à étudier les caractéristiques d'un nouvel objet pour l'attribuer à une classe prédéfinie.
Estimation	L'estimation est similaire à la classification à part que la variable de sortie est numérique plutôt que catégorique, En fonction des autres champs de l'enregistrement l'estimation consiste à compléter une valeur manquante dans un champ particulier
Groupement par similitude	Le groupement par similitude ou l'Analyse des associations et de motifs séquentiels consiste à déterminer quels attributs "vont ensemble"
Analyse des clusters	L'analyse de clusters ou Le clustering (ou la segmentation) est le regroupement d'enregistrements ou des observations en classes d'objets similaires. Tandis qu'Un cluster est une collection d'enregistrements similaires l'un à l'autre, et différents de ceux existants dans les autres clusters.
Description	Cette tâche consiste à décrire ce qui se passe sur une Base de Données compliquée en expliquant les relations existantes dans les Données pour en premier lieu comprendre le mieux possible les individus, les produit et les processus présents dans cette base.

Tableau 2.2 : Tâches d'Exploration de données**.**

2.3.4 Disciplines incorporées en Exploration de données

L'exploration de données a aujourd'hui une grande importance économique du fait qu'elle permet d'optimiser la gestion des ressources (humaines et matérielles), elle comporte plusieurs disciplines qui touchent directement notre vie pratique, ce qui démontre son utilité applicative. Ces disciplines impliquent de nombreux domaines de recherche, et exigent

l'existence et la disponibilité d'un ensemble de données de problème à étudier. Parmi ces disciplines nous pouvons citer [41] :

- L'apprentissage automatique.
- La reconnaissance des formes.
- Les statistiques.
- Les bases de données et entrepôts de données.
- La visualisation.
- La recherche d'information.

2.3.5 Exploration des données et data Lake

Il est important que des informations utiles puissent être extraites des lacs de données. Cependant, le grand nombre de source ingérées et l'hétérogénéité des données rendent cette tâche difficile à mettre en œuvre; un utilisateur peut avoir connaissance d'une ou de quelques sources de données, mais rarement de tous les ensembles de données, Ainsi les solutions existantes résolvent principalement le problème d'interrogation dans les lacs de données dans les directions suivantes [43] :

1. Découvrir les lacs de données en fonction de la parenté des ensembles de données.
2. Fournir une interface de requête unifiée pour les sources de données hétérogènes.

De côté d'utilisation des lacs de données, l'exploration des lacs de données à destination seulement de quelques utilisateurs, et son outillage technologique n'est pas accessible à des utilisateurs non avertis. En d'autres termes, les lacs de données sont à destination des spécialistes de la science : les « data scientists » [38]

2.3.5.1 Query-Driven data discovery

La première approche se concentre sur la découverte des données (data discovery). C'est un processus permettant la collecte et l'évaluation de données provenant de sources diverses. Elle se base sur deux approches, *l'exploration des données* et *l'analytique visuelle***.** Le Query-Driven data discovery ou découverte de données basée sur des requêtes fait référence à la recherche d'un lac de données en fonction de la relation mesurée entre les ensembles de données en formulant un ensemble des requêtes avec des données en entrées, et un ensemble de données les plus liés en sortis [43].

Méthode d'exploration

Nous allons présenter ci-dessous les trois méthodes d'exploration proposées en [43] en se basant sur les quatre systèmes de découverte de données **AURUM, JOSE, D^{3}L, JUNEAU** en notant que 'S'signifie l'ensemble de données stockées dans le lac de données.

- **Méthode 1 :** Étant donné la table T spécifiée par l'utilisateur et une colonne C de T, le système renvoie les tables Top-k qui sont les plus liées à T (cas de JOSIE).
- **Méthode 2 :** Étant donnée la table T spécifiée par l'utilisateur, le système renvoie les tables Top-k (S^k) contient les attributs pertinents pour remplir T. De plus si une table S_i n'est pas dans le jeu de résultat Top-k, mais il peut être joint à certains table(S) en S^k, et améliorer la couverture attributaire de T, alors S_i peut être inclue dans le résultat (cas de D^3L).
- **Méthode 3 :** Étant donné la table T spécifiée par l'utilisateur et le type de recherche pour les applications externes (par exemple, une tâche de science de données), alors le système renvoie les Top-k tables les plus pertinents pour T sur la base des mesures de relation associées à T (cas de JUNEAU).

  A noter que dans ces trois méthodes, l'exploration d'un ensemble de données est difficile dans la recherche plus que dans la reformulation de requête dans l'intégration de données.

Méthodes d'interrogation et d'indexation [43]

Généralement les lacs de données basé sur l'estimation de similarité à l'aide d'indexation dans une table de requête d'entrée, ils classent les tables candidats et incluent les tables Top-k dans l'ensemble de résultats, nous allons expliquer ci-dessus les méthodes d'indexation et d'interrogation utilisées par les deux systèmes Aurum, Juneau :

- Le système Aurum applique les indexes de graphes pour accélérer les requêtes coûteuses, contenant des requêtes de chemin de découverte pour rechercher ses hypergraphes.
- Le système Juneau considère une requête en tant qu'une table de sortie de cellule choisie par utilisateur. En d'autres termes, c'est l'utilisateur qui précise le type et le schéma de recherche. Ce système utilise les mesures de parenté correspondantes pour effectuer la recherche.

2.3.5.2 Query-heterogenous data [43]

La deuxième approche se concentre sur la fourniture d'une interface de requête unifiée pour les données hétérogènes, c'est-à-dire un utilisateur peut connaître et parcourir tous les sources de données existantes avec leur description, leurs statistiques et leurs schéma, puis il peut effectuer la recherche via l'interface utilisateur par des mot-clés sur les données ou le schéma ou avec des requêtes SQL, de plus les utilisateurs peuvent transforme les données du lac de données selon la structure et le format souhaitée.

2.4 Conclusion

Dans ce chapitre, nous avons présenté d'une façon détaillée la nouvelle technologie de stockage de Big Data qui s'appelle lacs de données (ou Data Lake). Nous avons également montré que l'exploitation de ces lacs de données nécessite une base vitale qui s'appelle les métadonnées. Le dernier aspect que nous avons introduit dans ce chapitre est le processus d'exploration des données avec ses techniques et ses phases qui permettent de générer des connaissances à partir de grands volume de données.

Chapitre 03 : Capture des besoins fonctionnels.

3.1 Introduction

Ce chapitre nous sert à projeter et détailler l'étape qui précède le processus de développement que nous allons utiliser dans notre projet, à savoir **le processus 2TUP** qui est **«** l'étude préliminaire **»** ainsi que sa première étape **«** la capture des besoins fonctionnels **»** de notre système. Dans un premier temps, nous allons introduire l'étude de cas qui servira de fil conducteur tout au long de notre travail, en donnant une définition et en expliquant son principe de fonctionnement. Dans un second temps, nous commencerons à détailler l'étape préliminaire (ou pré-étude) de notre travail, qui consiste à effectuer un premier repérage des besoins fonctionnels. Nous allons définir dans cette étape les limites (contour) du système à réaliser, les différents acteurs avec leurs responsabilités ; ainsi que les messages d'interaction entre les acteurs et le système. Ensuite, nous allons détailler la phase de capture des besoins fonctionnels, qui formalise ce qui a été ébauché au cours de l'étude préliminaire. Durant cette étape, nous allons identifier et décrire l'ensemble des cas d'utilisation, élaborer un diagramme global de cas d'utilisation et identifier pour chaque cas l'ensemble de ses classes candidates.

3.2 Processus de développement

2TUP signifie **« 2 Track Unified Process »,** c'est un processus de développement logiciel qui implémente le processus unifié ; et qui répond aux caractéristiques que nous venons de citer. Le processus 2TUP apporte une réponse aux contraintes de changement continuel imposées aux systèmes d'information de l'entreprise. En ce sens, il renforce le contrôle sur les capacités d'évolution et de correction de tels systèmes. Le 2TUP propose un cycle de développement en Y qui dissocie les aspects techniques des aspects fonctionnels, dans une suite des étapes qui sont précédées par une étape de prise de contact avec le domaine et l'environnement de l'entreprise appelée Étude préliminaire. La figure 3.1 représente les différentes étapes du processus 2TUP [45] :

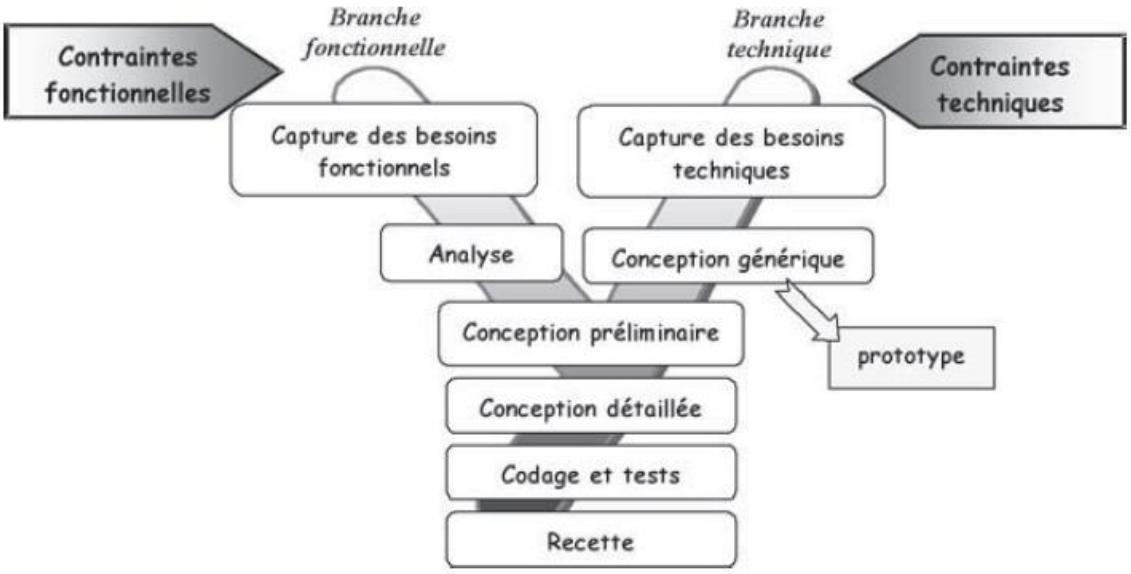


Figure 3.1 : Le processus de développement 2TUP [45].

Dans notre système nous appliquons le processus 2TUP d'une manière allégée, en n'utilisant que les étapes qui servent les spécificités de notre travail et qui sont la capture des besoins fonctionnels, l'analyse et la conception. L'adaptation du processus 2TUP est schématisée en figure 3.2

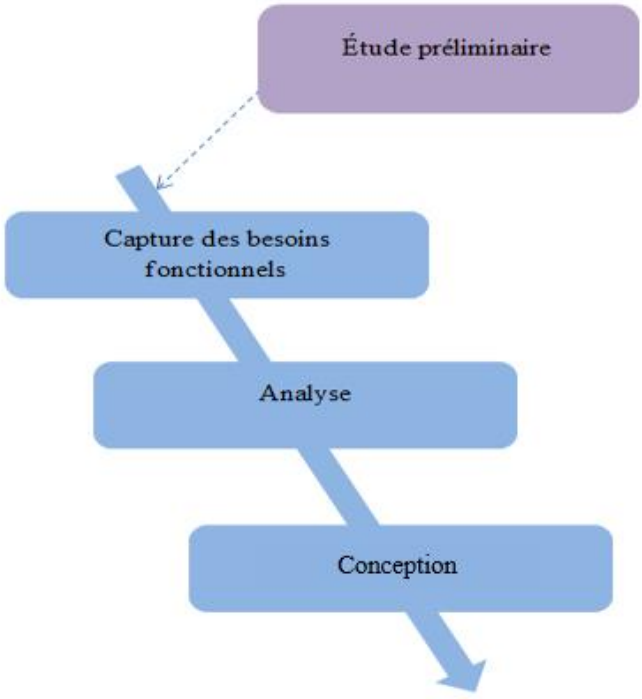


Figure 3.2 : Processus 2TUP allégé.

3.3 Étude de cas : Réseau d'Entraide « StackExchange »

Dans le cadre de notre travail qui se base sur l'exploration d'un Data Lake alimenté par différents types de ressources de données, il est nécessaire d'avoir un cas d'étude qui sert d'exemple courant de monde réel pour illustrer les différentes étapes de notre étude et montrer

son applicabilité. Pour cela nous allons baser notre travail sur un système de type Question-Réponse qui s'appelle StackExchange.

3.3.1 Qu'est-ce que le Réseau StackExchange ?

Le réseau StackExchange est un réseau des sites web de questions- réponses sur des sujets dans des divers domaines; chaque site couvre un sujet spécifique où les questions, les réponses et les utilisateurs sont soumis à un processus d'attribution de réputation. Ce réseau est créé par Joël Spolsky et Jeff Atwood en août 2008 en tant qu'un site commercial en anglais [44].

3.3.2 Fonctionnement du réseau Stackexchange

Nous avons perçu la réalité et le comportement de réseau StackExchange comme étant composé d'**entités** qui interagissent entre elles sous forme des **relations**. Le réseau est composé d'**Users** (utilisateurs de réseau), **Comments** (commentaires des utilisateurs), **Posts** (les publications des utilisateurs), **Votes** (des votes sur des postes), **Tags** (balises sur les postes), **Badges** (insigne bronze, argent ou or), et des **Flags** (marquages pour le contenu de postes). Les postes, les votes et les flags peuvent avoir des différents types.

Les utilisateurs peuvent poser, éditer, répondre aux questions, soumettre des réponses, commenter un ou plusieurs postes d'autres utilisateurs. Par la suite, ils peuvent voter pour les questions et les réponses qui leurs semblent pertinentes soit positivement (Vote Up) ou négativement (Vote Down); ces actions de votes permettent de faire gagner des points appelés réputation, et influencer la visibilité des messages: une réponse ou une question avec le plus de votes positifs sera affichée en haut de page et les réponses et les questions avec aucun vote, voire des votes uniquement négatifs sont reléguées en bas de page [44][45].

Pour le besoin de notre travail, et pour illustrer l'exploration d'un lac de données, nous avons implémenté un lac de données de réseau StackExchange, composé de trois sources de données: relationnelle, document et graphe, tandis que le réseau StackExchange n'est qu'un exemple courant pour parcourir les étapes de notre étude.

3.4 Étude préliminaire

3.4.1 Identification des besoins fonctionnels

Les besoins fonctionnels de notre système sont décrits ci-dessus, en proposant pour chaque besoin un exemple à partir le réseau Stackexchange:

[3]Dans l'identification des besoins, on décrit par **« composant »** tous qui rentrent dans la composition d'une structure (qui compose structurellement une source de données), par exemple : une source relationnelle est composée de tables, une source graphe est composée de nœuds et des Relationships, et une source document est composée de collections (Voir chapitre 1).

a. Connaître les différents types de sources qui alimentent le lac de données (comment les données des entités et leurs relations sont stockées dans le lac de données). **Par exemple :** afficher les types de sources de données de Stackexchange Data Lake (sources relationnelles, sources graphes, sources documents).

b. Savoir pour chaque source quels sont ses composants stockés dans le lac de données.

Par exemple : afficher la liste des Tables relationnelles (source relationnelle), la liste des Nœuds et Relationship graphes (source graphe), la liste des Collections documents (source document).

c. Avoir une vue complète (360°) d'une entité du monde réel.

Par exemple : Avoir une vue complète sur les utilisateurs (Afficher toutes les informations concernant les utilisateurs quelle que soit leur nature).

d. Connaître les informations structurelles qui décrivent un composant d'une structure donnée.

Par exemple : Afficher les colonnes d'une table relationnelle, les propriétés d'un nœud ou d'une relation entre nœuds d'un graphe, les items d'une collection document.

e. Connaître les liens entre les entités sémantiques.

Par exemple : Afficher toutes les informations concernant la relation de publication (un utilisateur publie des postes).

f. Accès aux données d'une entité

- Accès aux données d'une entité structurée. **Par exemple :** Afficher les données de la table relationnelle « Users ».
- Accès aux données d'une entité document. **Par exemple :** Afficher les items de la collection « Users ».
- Accès aux données d'une entité graphe. **Par exemple :** Afficher les nœuds de label « Users ».

g. Accès aux données d'une relation

- Accès aux données d'une relation relationnelle, i e. un lien entre tables.
- Accès aux données d'une relation graphe.
- Accès aux données d'une relation document.

h. Accès aux données d'une entité donnée par un mot clé ou une spécification métier (Business Name) avec la possibilité de synchroniser ces données à 360° pour une entité donnée avec tous les types de sources qui la décrivent.

Par exemple : Afficher les données des utilisateurs avec les trois types de sources (relationnelle, graphe, document).

i. Connaître les éléments contenus dans le lac de données pour un type de données particulier, ou d'une propriété de métadonnées particulière.

Par exemple : Afficher les colonnes des tables, les propriétés des nœuds, les items des documents ayant le type de données « date ».

j. Connaître les métadonnées d'un élément du Data Lake.

Par exemple : Afficher les métadonnées d'une table relationnelle (les colonnes, le type de données, le nom métier ou *Business Name*, la description, les contraintes...etc.), d'un nœud de graphe (les propriétés, le type de données, le Business Name, la description, les contraintes...etc.), d'une collection de document (les items, le type de données, le Business Name, la description, les contraintes...etc.)

3.4.2 Identification des acteurs

Notre système d'exploration de données comprend un seul type d'acteur : **l'Explorateur du lac de données** qui accède, découvre et visualise les données.

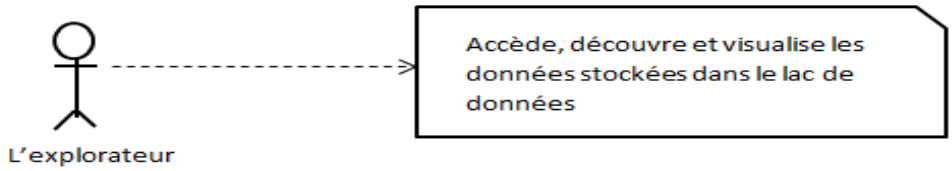
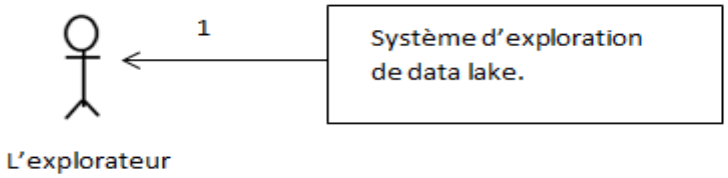


Figure 3.3 : Responsabilités des acteurs du système.

3.4.3 Modélisation du contexte

La figure 3.4 représente le diagramme de contexte dynamique de notre système.



Numéro de message	Nature de message
1	Données et métadonnées de DL.

Figure 3.4 : Diagramme de contexte dynamique.

3.5 Capture des besoins fonctionnels

3.5.1 Identification et description des cas d'utilisations

A partir de l'analyse des besoins fonctionnels, nous avons exprimé la description des cas d'utilisation de notre système:

- **Cas 1 :** Consulter la liste des types de sources de DL.

  But : Permet aux explorateurs de connaître les différents types de sources qui existent dans le DL.

  Précondition : le DL doit contenir au moins une source de données.

  Enchaînements :

  - L'Explorateur demande la liste des types de sources de DL.
  - Le Système affiche la liste des types de sources de DL.

- **Cas 2 :** Consulter la liste des sources d'un type de source de données de DL (par exemple : les sources relationnelles seulement).

  But : Permet aux explorateurs de connaître la liste des sources existantes pour un type de sources donné.

  Précondition : L'existence d'au moins une source pour le type de source choisi.

  Enchaînements :

  - L'Explorateur choisit un type de source de données de DL.
  - Le système affiche la liste des sources de type choisi par l'explorateur.

- **Cas 3 :** Consulter la structure d'un élément de DL.

  But : Permet aux explorateurs de connaître la structure d'un élément de DL choisi.

  Précondition : /

  Enchaînements :

  - L'Explorateur choisit un élément de DL.
  - Le système affiche la structure d'élément de DL choisi.

- **Cas 4 :** Consulter la liste des tables relationnelles d'une source relationnelle de DL.

  But : Permet aux explorateurs de connaître la liste des tables relationnelles d'une source relationnelle choisie.

  Précondition: l'existence d'au moins une table relationnelle dans la source relationnelle choisie.

  Enchaînements :

  - L'Explorateur choisit une source relationnelle de DL.
  - Le système affiche la liste des tables relationnelles de la source choisie par l'explorateur.

- **Cas 5 :** Consulter la liste des colonnes d'une table relationnelle.

  But : Permet aux explorateurs de connaître la liste des colonnes d'une table relationnelle choisie.

  Précondition : \

Enchaînements :

- L'Explorateur choisit une table relationnelle.
- Le système affiche la liste des colonnes de la table relationnelle choisie par l'explorateur.

- **Cas 6 :** Consulter La liste des références relationnelles (entre les colonnes) d'une source relationnelle.

  But : permet aux explorateurs de connaître la liste des liens qui existent entre les colonnes dans une source relationnelle.

  Précondition : L'existence d'au moins une référence relationnelle dans la source relationnelle.

  Enchaînements :

  - L'Explorateur choisit une source relationnelle de DL.
  - Le système affiche la liste des références relationnelles de la source choisie par l'explorateur.
- **Cas 7 :** Consulter la liste des collections d'une source document de DL.

  But : Permet aux explorateurs de connaître la liste des collections d'une source document choisie.

  Précondition : l'existence d'au moins une collection document dans la source document choisie.

  Enchaînements :

  - L'Explorateur choisit une source de documents de DL.
  - Le système affiche la liste des collections de la source choisie par l'explorateur.
- **Cas 8 :** Consulter la liste des items d'une collection de document.

  But : Permet aux explorateurs de connaître la liste des items de premier niveau d'une collection document choisie.

  Précondition : \

  Enchaînements :

  - L'explorateur choisit une collection de documents.
  - Le système affiche les items de la collection choisie par l'explorateur.
- **Cas 9 :** Consulter la liste des items composants d'un item particulier.

  But : Permet aux explorateurs de connaître la liste des items composants d'un item document choisi (s'ils existent).

Précondition : \

Enchaînements :

- L'explorateur choisit un item particulier d'une collection de document.
- Le système affiche la liste des items composants de l'item choisi.

- **Cas 10 :** Consulter la liste des références documents (entre les items) d'une source document.

  But : permet aux explorateurs de connaître la liste des liens qui existent entre les items dans une source document.

  Précondition : L'existence d'au moins une référence document dans la source document.

  Enchaînements :

  - L'Explorateur choisit une source de documents de DL.
  - Le système affiche la liste des références documents de la source choisie par l'explorateur.

- **Cas 11 :** Consulter la liste des Vertices (nœuds) ou Edges (liens) d'une source graphe.

  But : Permet aux explorateurs de connaître la liste vertices ou edges d'une source graphe choisie.

  Précondition : L'existence d'au moins un vertex ou une edge dans la source graphe choisie.

  Enchaînements :

  - L'explorateur choisit une source graphe de DL.
  - Le système affiche la liste des vertices et edges de la source graphe choisie.

- **Cas 12 :** Consulter la liste des propriétés d'un Vertex ou d'une Edge.

  But : Permet aux explorateurs de connaître la liste des propriétés d'un vertex ou d'un Edge choisi.

  Précondition : \

  Enchaînements :

  - L'explorateur choisit un vertex ou une edge.
  - Le système affiche les propriétés de vertex ou edge choisie par explorateur.

- **Cas 13 :** Accès aux données d'une source graphe.

  But : Permet aux explorateurs d'accéder aux données d'une source graphe choisie.

  Précondition : /

[4] Un vertex signifie un nœud d'un graphe, et une Edge signifie une arête (lien) entre deux nœuds. On utilise ces termes pour les différencier des nœuds et liens du graphe des métadonnées.

Enchaînements :

- L'explorateur choisit une source graphe.
- Le Système affiche les données de la source graphe choisie.

- **Cas 14 :** Accès aux données.

  But : Permet aux explorateurs d'accéder aux données d'un élément de DL choisi.

  Précondition : /

  Enchaînements :

  - L'explorateur choisit un élément de DL.
  - Le Système affiche les données de l'élément choisi.
- **Cas 15 :** Accès aux métadonnées.

  But : Permet aux explorateurs d'accéder aux métadonnées d'un élément de DL choisi.

  Précondition : /

  Enchaînements :

  - L'explorateur choisit un élément de DL.
  - Le Système affiche les métadonnées de l'élément choisi.
- **Cas 16 :** Consulter la liste des entités et liens sémantiques (comprenant toutes les sources de DL).

  But : Permet aux explorateurs d'aligner sémantiquement les éléments de DL.

  Précondition : /

  Enchaînements :

  - L'explorateur choisit le type (élément) d'alignement.
  - Le système affiche la liste des éléments alignés sémantiquement avec le type choisi par l'explorateur.
- **Cas 17 :** Rechercher un élément de DL.

  But : permet aux explorateurs de rechercher un élément de DL en entrant des mots-clé ou des spécifications (tels que les propriétés des métadonnées : Business Name, type de données etc. …)

  Pré condition : /

  Enchaînements :

  - L'explorateur saisit le nom d'un éventuel élément de DL par un mot clé.
  - Le système retourne tous les éléments de DL correspondants au mot-clé entré par l'explorateur.

3.5.2 Élaboration du diagramme des cas d'utilisations

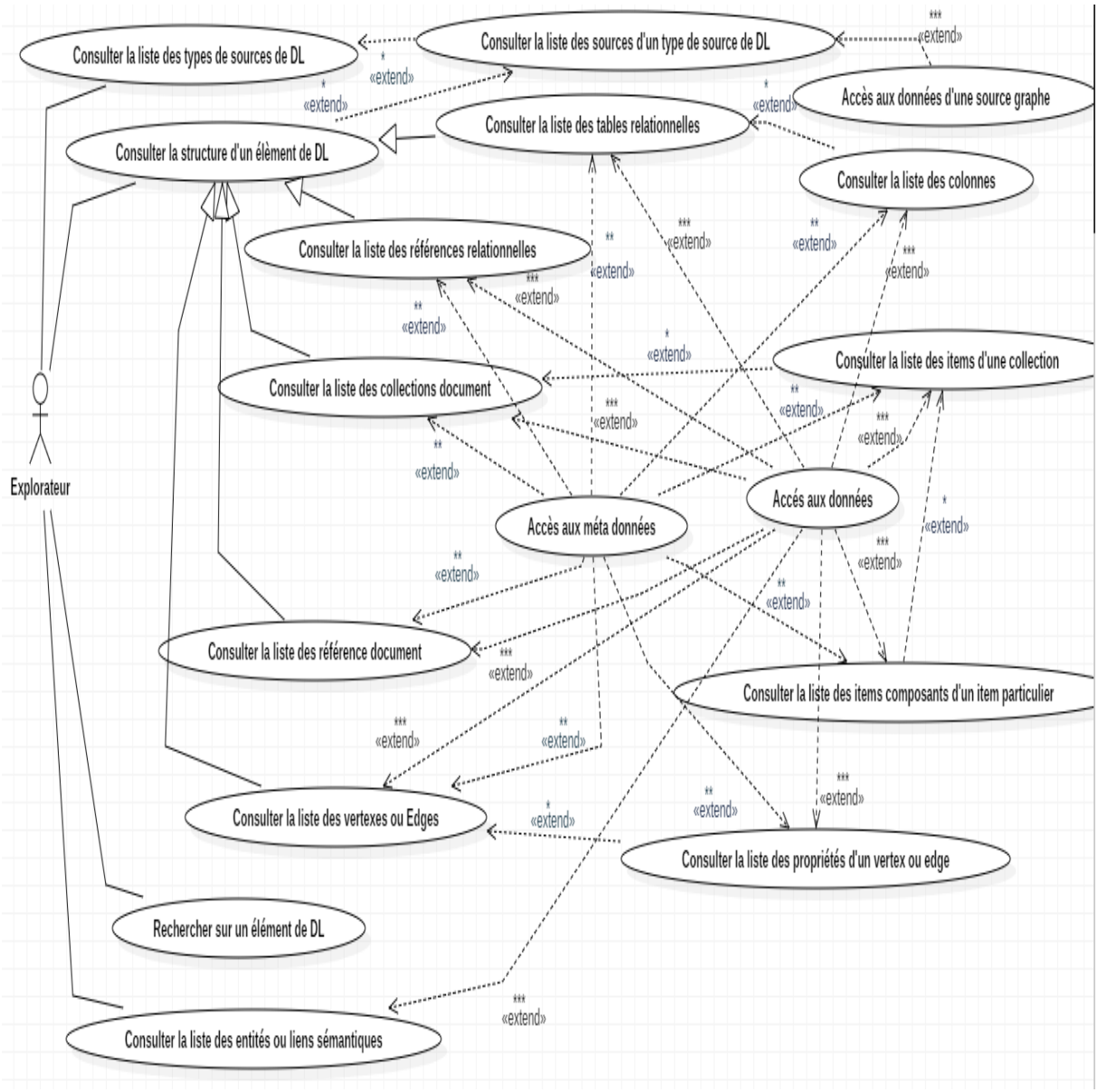


Figure 3.5: Diagramme de cas d'utilisation de notre système.

Dans le diagramme de cas d'utilisation :

- extend * : signifie lien d'extension avec le points d'extension de « besoin de détailler la structure » .
- extend ** : signifie lien d'extension avec le points d'extension de « besoin d'accéder aux données ».
- extend *** : signifie lien d'extension avec le points d'extension « besoin d'accéder aux métadonnées ».

3.5.3 Identification des classes candidates

Cette étape consiste à identifier la liste préliminaire des classes qui permettent de répondre aux exigences statiques (attributs) et dynamiques (opérations) de chaque cas d'utilisation.

- **Cas d'utilisation «** Consulter la liste des types de sources de DL » : les classes candidates de ce cas sont **Dl** ,**Rel_DS**, **Gph_DS**, **Doc_DS** et **MD_prp** (voir figure 3.6).

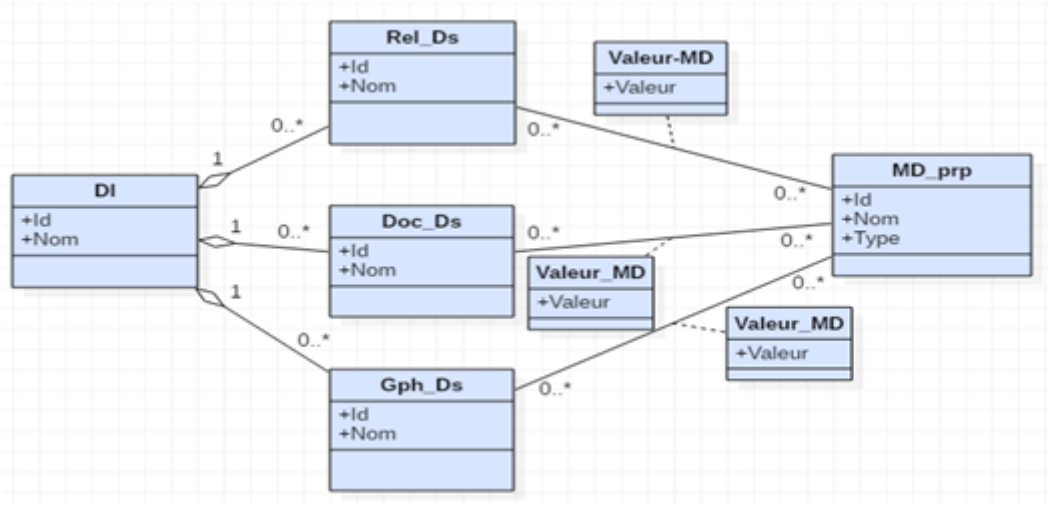


Figure 3.6: Diagramme de classes candidates de cas d'utilisation « Consulter la liste des types de sources de DL».

- **Cas d'utilisation «** Consulter la liste des sources d'un type de source de données de DL» : les classes candidates de ce cas sont **Dl** ,**Rel_DS**, **Gph_DS**, **Doc_DS** et **MD_prp** (voir figure 3.7).

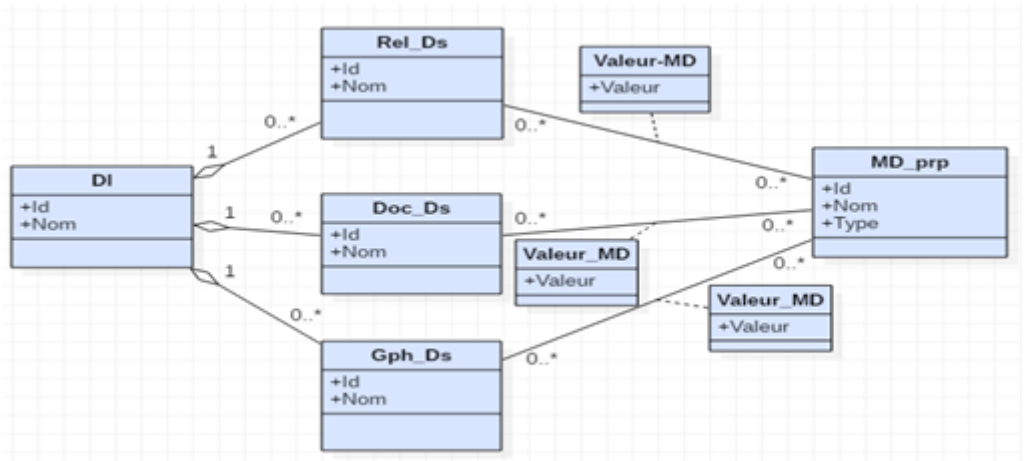


Figure 3.7 : Diagramme de classes candidates de cas d'utilisation « Consulter la liste des sources d'un type de source de données de DL ».

- **Cas d'utilisation «** Consulter la liste des tables relationnelles d'une source relationnelle de DL » : les classes candidates de ce cas sont **Rel_Tab** et **Rel_DS** (voir figure 3.8).

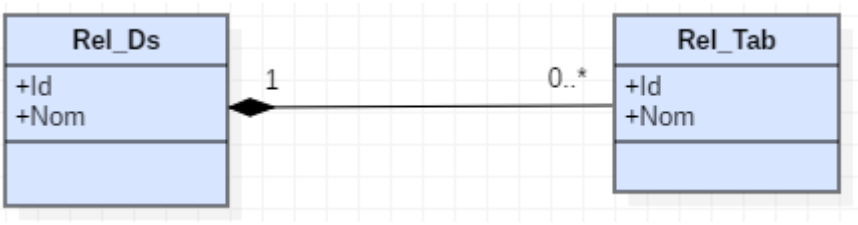


Figure 3.8 : Diagramme de classes candidates de cas d'utilisation « Consulter la liste des tables relationnelles d'une source relationnelle de DL».

- **Cas d'utilisation «** Consulter la liste des colonnes d'une table relationnelle» : les classes candidates de ce cas sont **Rel_Tab** et **Rel_Col** (voir figure 3.9).

Figure 3.9 : Diagramme de classes candidates de cas d'utilisation « Consulter la liste des colonnes d'une table relationnelle».

- **Cas d'utilisation «** Consulter la liste des références relationnelles d'une source relationnelle» : les classes candidates de ce cas sont **Rel_DS, Rel_Relationship** et **Rel_Col et Rel_Tab** (voir figure 3.10).

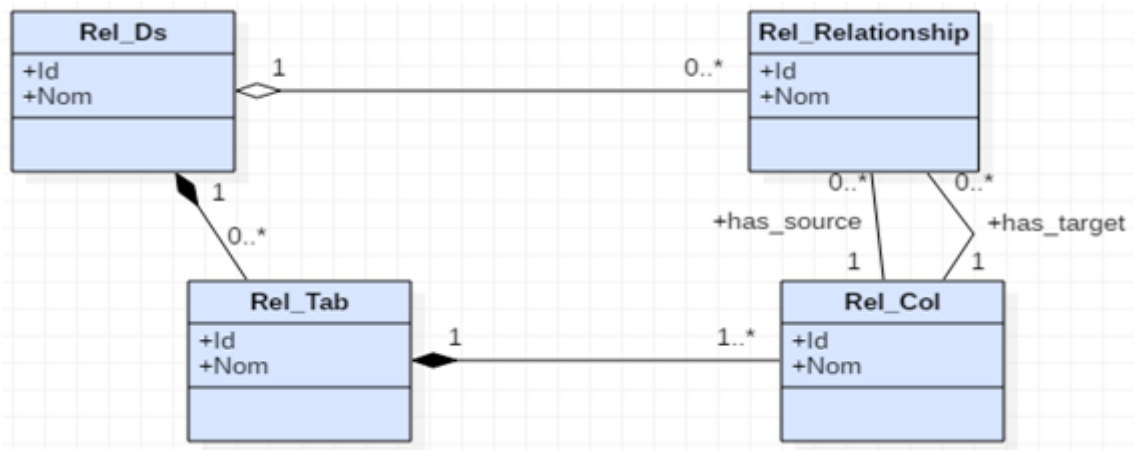


Figure 3.10 : Diagramme de classes candidates de cas d'utilisation « Consulter La liste des références relationnelles d'une source relationnelle».

- **Cas d'utilisation «** Consulter la liste des collections d'une source document de DL» : les classes candidates de ce cas sont **Doc_DS**, **et Doc_Col** (voir figure 3.11).

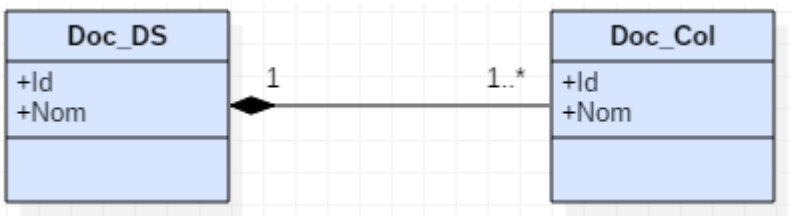


Figure 3.11 : Diagramme de classes candidates de cas d'utilisation « Consulter la liste des collections d'une source document de DL».

- **Cas d'utilisation «** Consulter la liste des items d'une collection document» : les classes candidates de ce cas sont **Doc_Col** et **Doc_Item** (voir figure 3.12).

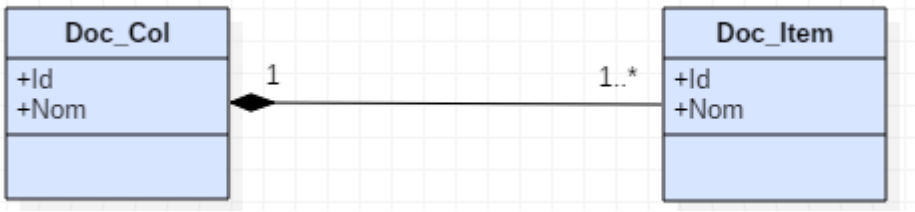


Figure 3.12 : Diagramme de classes candidates de cas d'utilisation « Consulter la liste des items d'une collection document ».

- **Cas d'utilisation «** Consulter la liste des items composants d'un item particulier» : les classes candidates de ce cas sont **Doc_Col** et **Doc_Item** (voir figure 3.13).

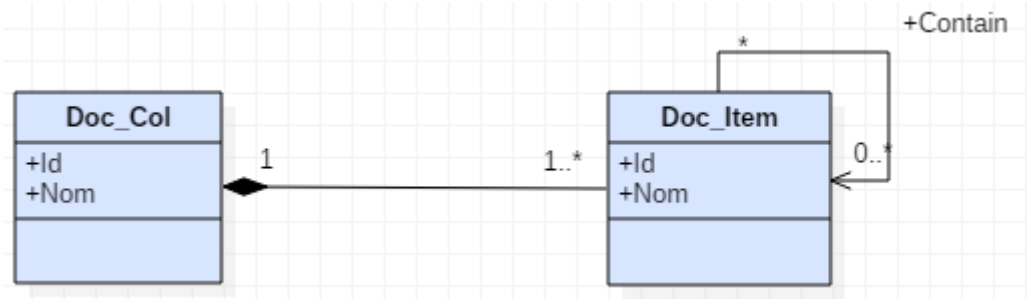


Figure 3.13 : Diagramme de classes candidates de cas d'utilisation « Consulter la liste des items composants d'un item particulier».

- **Cas d'utilisation «**Consulter la liste des références documents d'une source document» : les classes candidates de ce cas sont **Doc_DS**, **Doc_Relationship, Doc_Col** et **Doc_Item** (voir figure 3.14).

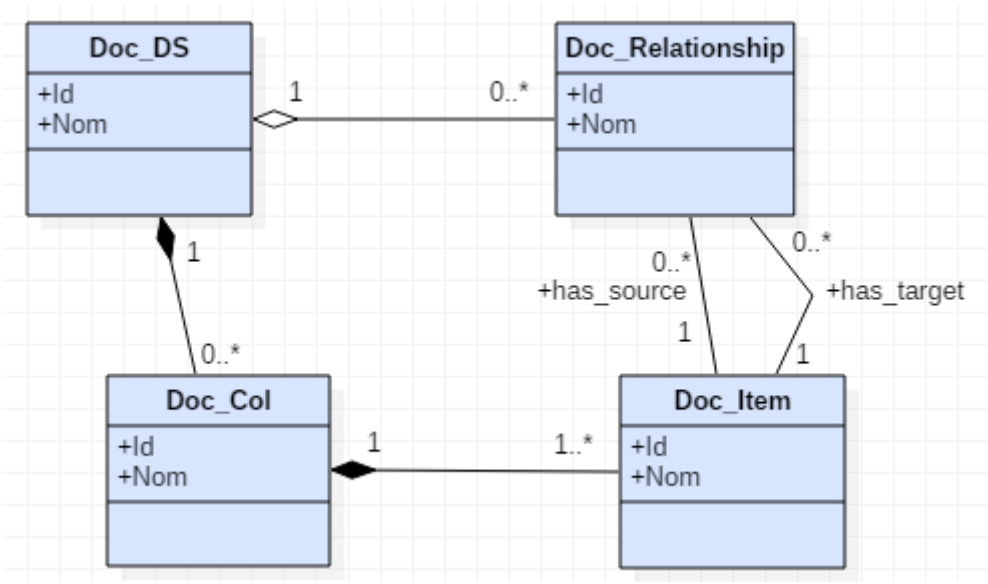


Figure 3.14 : Diagramme de classes candidates de cas d'utilisation « Consulter la liste des références documents d'une source document ».

- **Cas d'utilisation «** Consulter la liste des Vertices (Nodes) ou Edges (Relationships) d'une source graphe.» : les classes candidates de ce cas sont **Gph_DS ,Gph_Vtx et Gph_Edge** (voir figure 3.15).

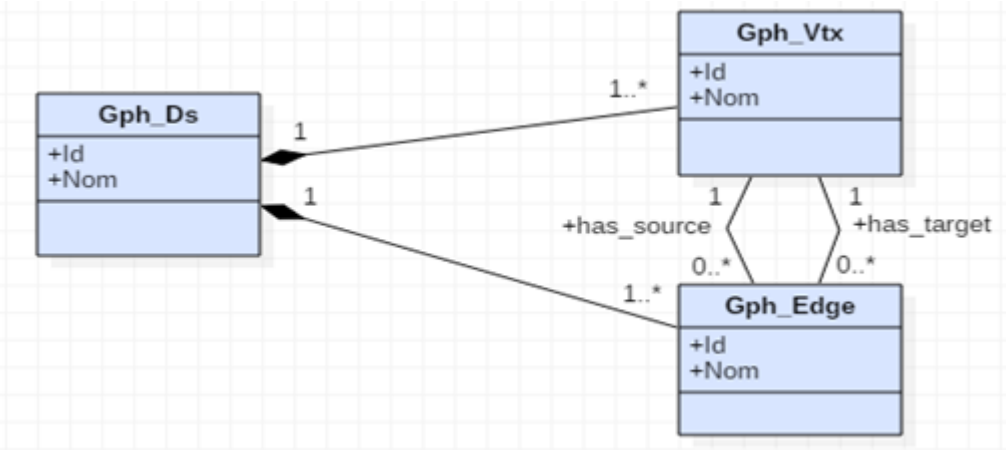


Figure 3.15 : Diagramme de classes candidates de cas d'utilisation « Consulter la liste des Vertices (Nodes) ou Edges (Relationships) d'une source graphe.».

- **Cas d'utilisation «** Consulter la liste des propriétés d'un Vertex ou d'un Edge» : les classes candidates de ce cas sont **Gph_Vtx, Gph_Edge et Gph_Prp** (voir figure 3.16).

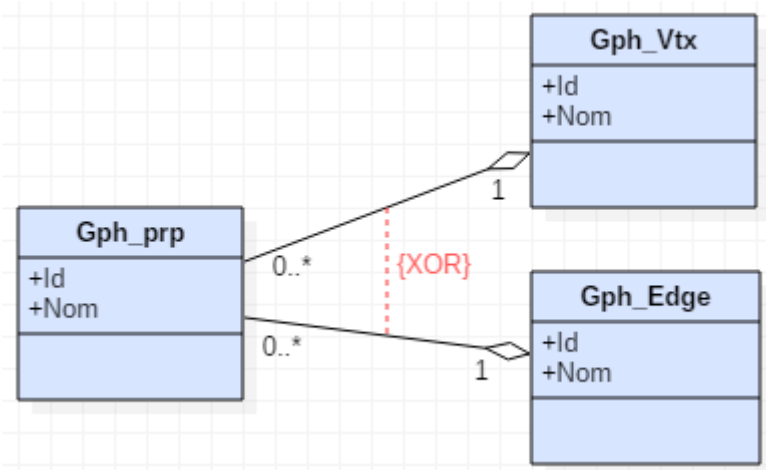


Figure 3.16 : Diagramme de classes candidates de cas d'utilisation « Consulter la liste des propriétés d'un Vertex ou d'un Edge».

- **Cas d'utilisation «** Accès aux données d'une source graphe» : les classes candidates de ce cas sont **Gph_DS**(voir figure 3.17).

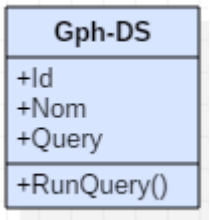


Figure 3.17: Diagramme de classes candidates de cas d'utilisation « Accès aux données d'une source graphe».

- **Cas d'utilisation** « Accès aux données»: Rel_Tab ,Rel_Col , Rel_Relationship ,Doc_Col , Doc_Item , Doc_Relationship ,Gph_Ds , Gph_Vtx , Gph_Edge , Gph_Prp (Voir figure 3.18).

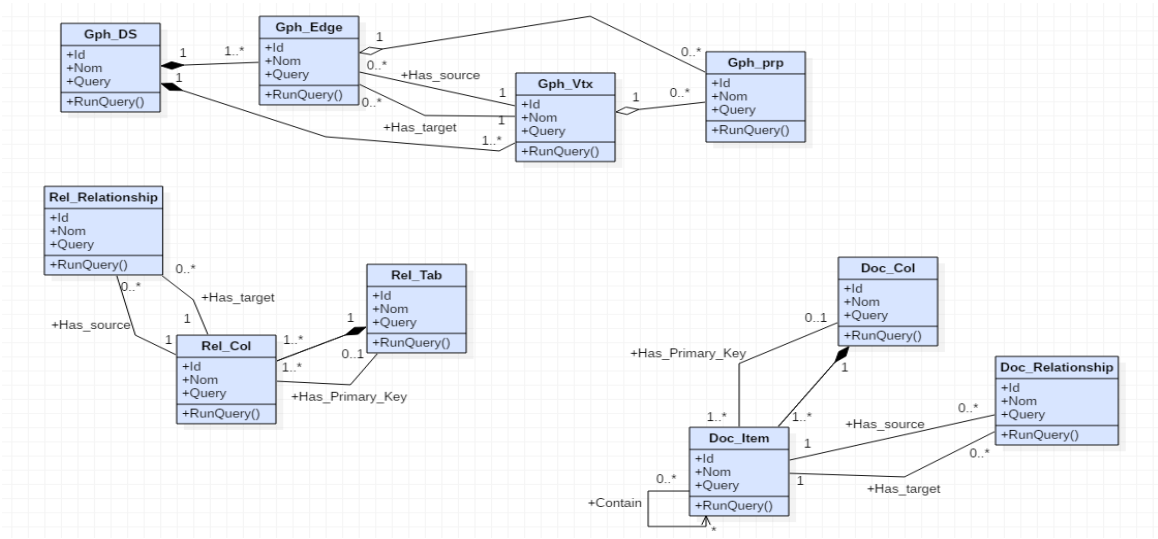


Figure 3.18 : Diagramme de classes candidates de cas d'utilisation « Accès aux données».

- **Cas d'utilisation «** Accès aux métadonnées» : les classes candidates de ce cas sont **MD_prp** et une classe générique **Nom_élément** qui représente n'importe quel élément ou classe de DL **(par exemple :** Dl, Gph_DS, Rel_Tab, Doc_Item, Rel_Relationshipect**)** (Voir figure 3.19).

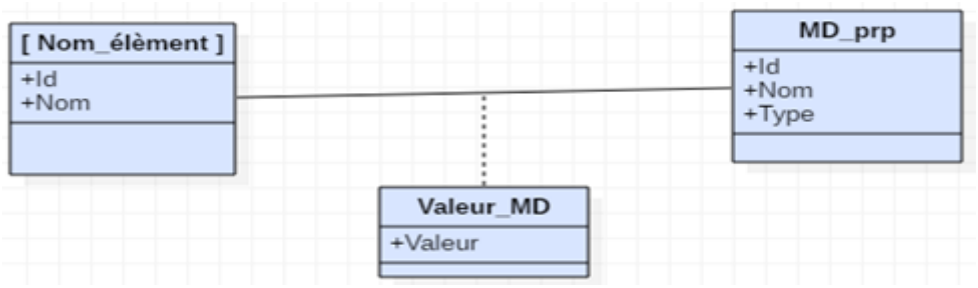


Figure 3.19 : Diagramme de classes candidates de cas d'utilisation « Accès aux Métadonnées ».

- **Cas d'utilisation «** Consulter la liste des entités ou liens sémantiques (avec toutes les sources de DL)» : les classes candidates de ce cas sont **DL , Gph_DS, Rel_DS, Doc_DS ,Gph_Edge , Gph_Vtx , Rel_Tab , Rel_Relationship , Doc_Col , Doc_Relationship , MD_prp** (Voir figure 3.20).

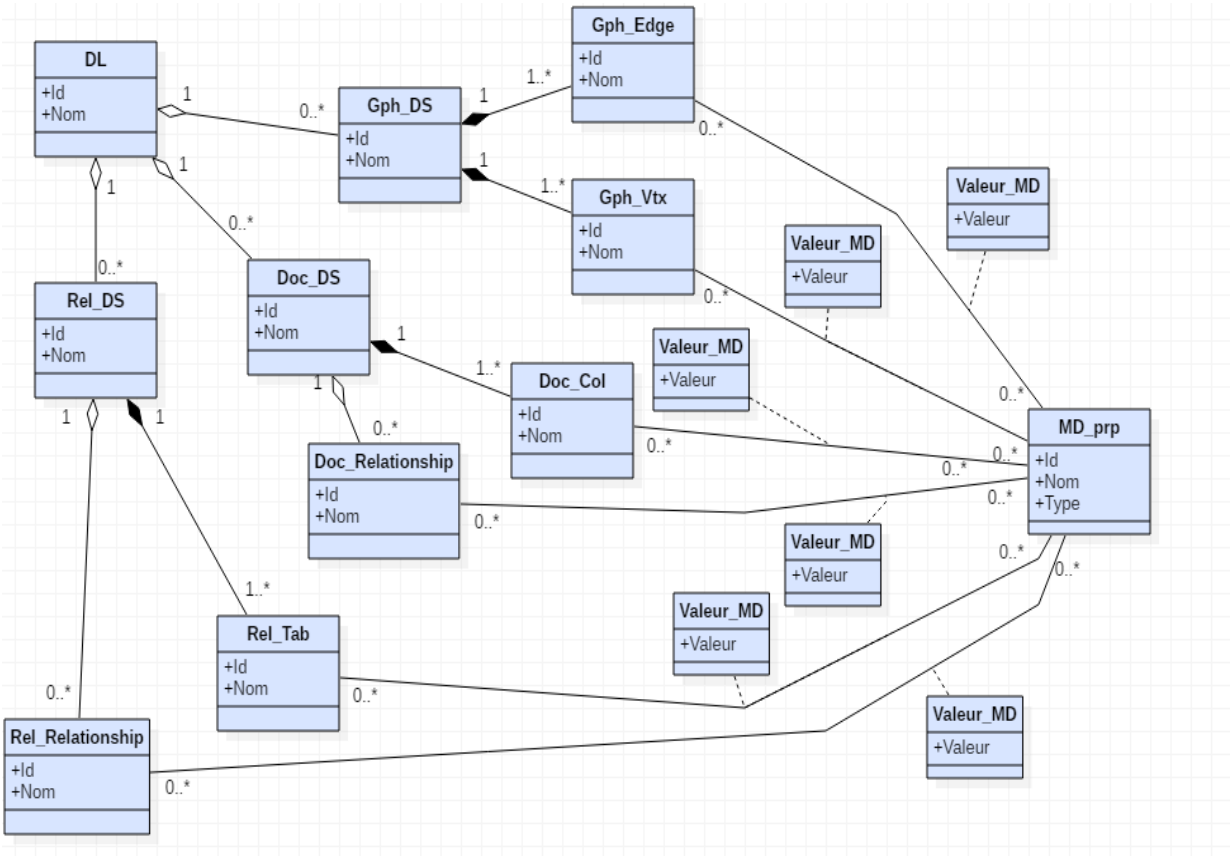


Figure 3.20 : Diagramme de classes candidates de cas d'utilisation « Consulter la liste des entités ou liens sémantiques ».

- **Cas d'utilisation «** Rechercher un élément de DL » : les classes candidates de ce cas sont **toutes les classes du système** (Voir figure 3.21).

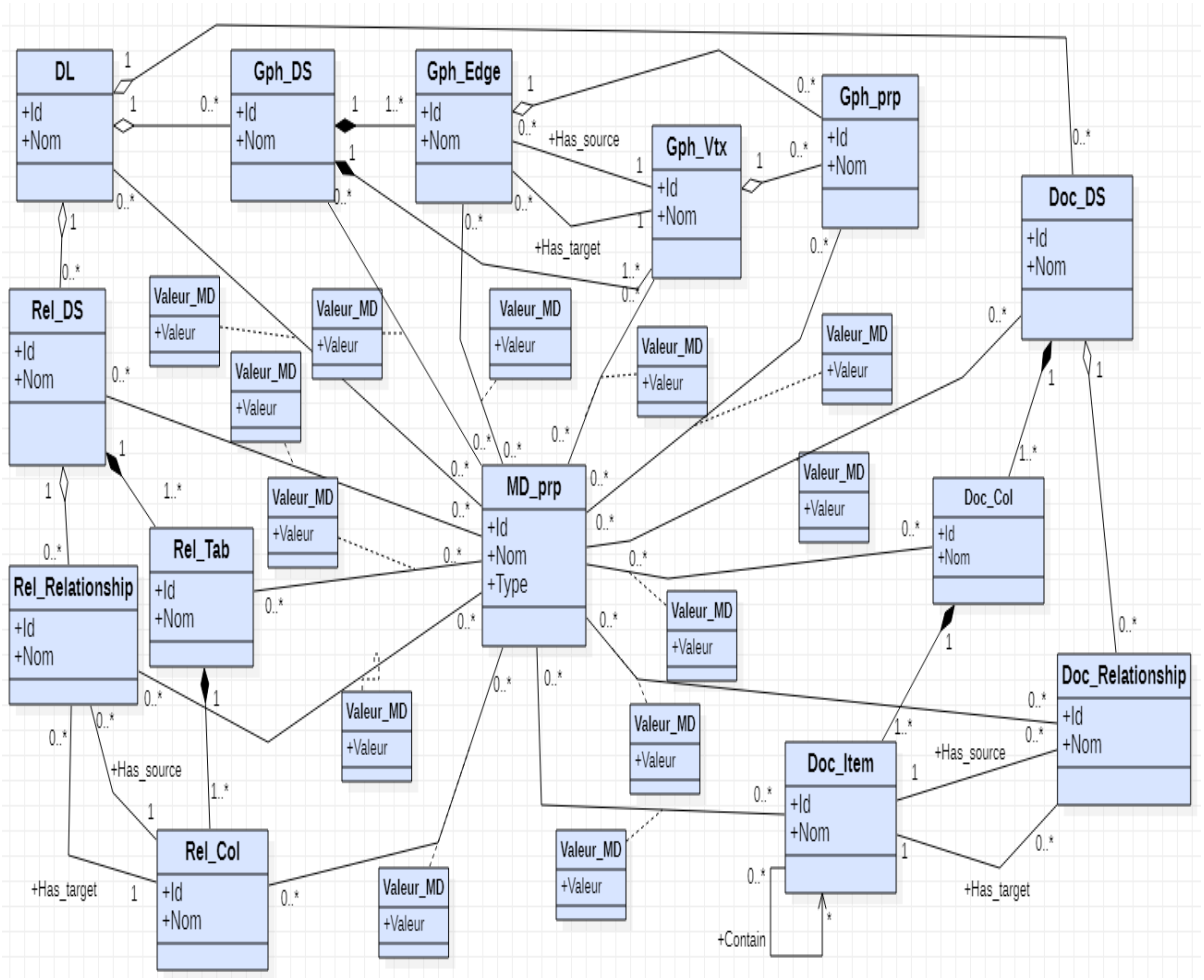


Figure 3.21 : Diagramme de classes candidates de cas d'utilisation « Rechercher un élément de DL »

3.6 Conclusion

Dans ce chapitre nous avons présenté les deux premières étapes du processus 2TUP : l'étude préliminaire et la capture des besoins fonctionnels de notre travail, où nous avons identifié dans l'étude préliminaire l'ensemble des besoins métiers du notre système, ainsi que les acteurs et les messages d'interaction avec le système. Ensuite dans la capture des besoins fonctionnels nous avons proposé et identifié les cas d'utilisation avec leur diagramme global et leurs classes candidates.

Chapitre 04 : Analyse et Conception

4.1 Introduction

Ce chapitre traite du démarrage de l'analyse objet du notre système, ainsi que sa conception. Dans un premier temps, nous détaillons l'étape de l'analyse de notre travail; qui permet d'approfondir la capture des besoins fonctionnels en détaillant l'aspect *données* par le développement du modèle statique et l'aspect *traitements* en développant l'aspect dynamique. Ensuite, nous continuons avec la phase de conception en détaillant l'étape de conception préliminaire du processus 2TUP nécessaire à notre travail. Cette étape consiste à tenir en compte des besoins exprimés dans la phase de capture des besoins fonctionnels dans la conception du logiciel; elle est certainement l'étape la plus délicate du processus 2TUP car elle en représente le cœur.

4.2 Analyse

4.2.1 Diagramme de classe global

Le développement du modèle statique comporte différentes étapes: découpage en catégories, traitement des anomalies, et déduction du diagramme global de classes qui sera à la base du développement du modèle dynamique. Dans notre travail, vu le nombre limité de classes (moins de 20 classes) ainsi que le fait que le travail n'est pas réalisé par des équipes distinctes, nous limitons le développement du modèle statique à la présentation du diagramme de classes global. La figure 3.1 représente le diagramme de classe global de notre système. Notons que la classe d'association **Valeur_MD** doit être dupliquée dans tous les liens entre la classe MD_prp et les autres classes, mais pour des raisons de clarté, nous la représentons une seule fois entre la classe DL et la classes MD_prp.

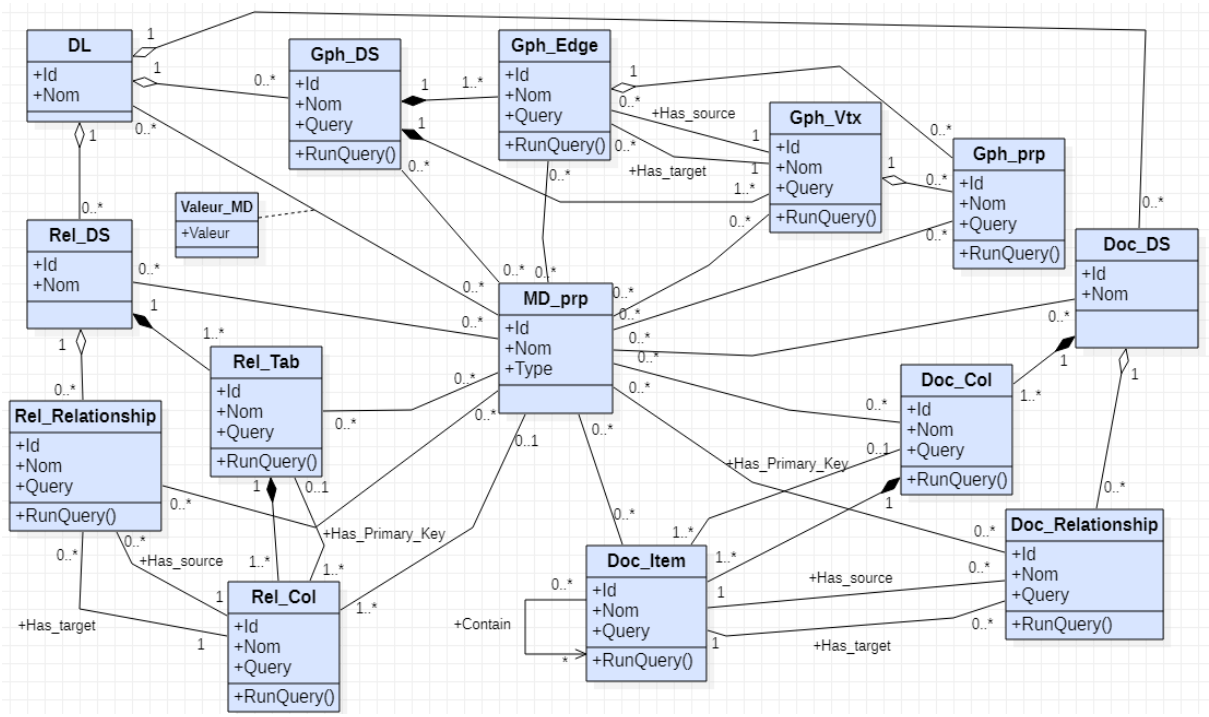


Figure 4.1 : Diagramme de classe globale de notre système.

4.2.2 Diagrammes de séquence

Dans cette section, nous présentons le développement du modèle dynamique en détaillant l'interaction entre les classes et les acteurs. Nous présentons chaque cas d'utilisation sous la forme de son scénario principal ou en illustrant les quelques scénarios d'exécution.

- **Cas d'utilisation** « Consulter la liste des types de sources de DL » (voir figure 4.2).

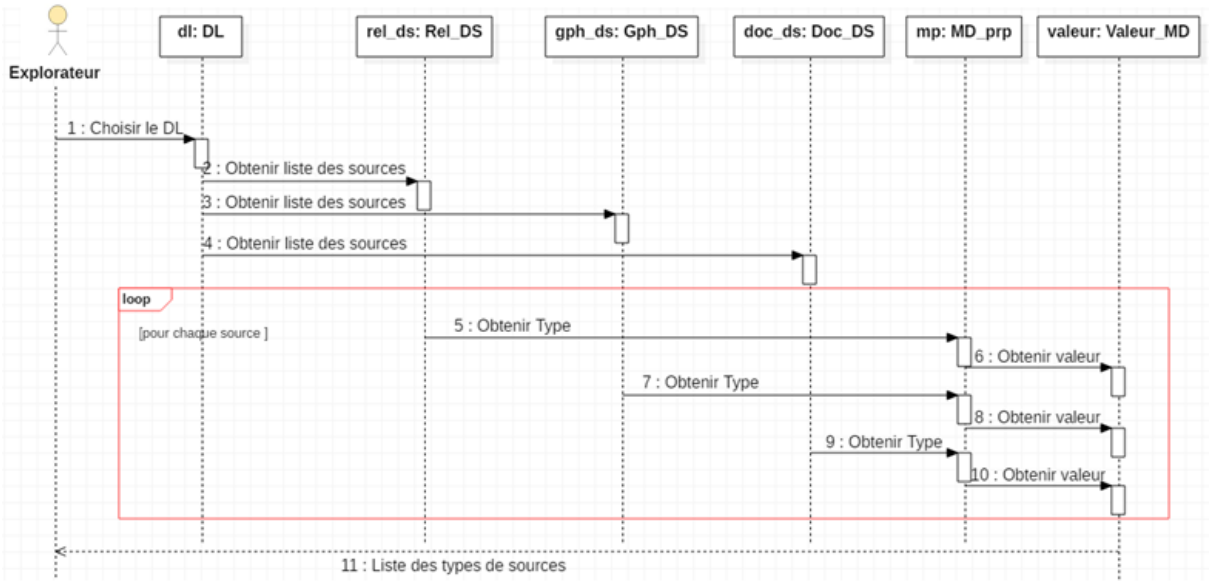


Figure 4.2 : Diagramme de séquence de cas d'utilisation « Consulter la liste des types de sources de DL ».

- **Cas d'utilisation** « Consulter la liste des sources d'un type de source de données de DL » (voir figure 4.3).

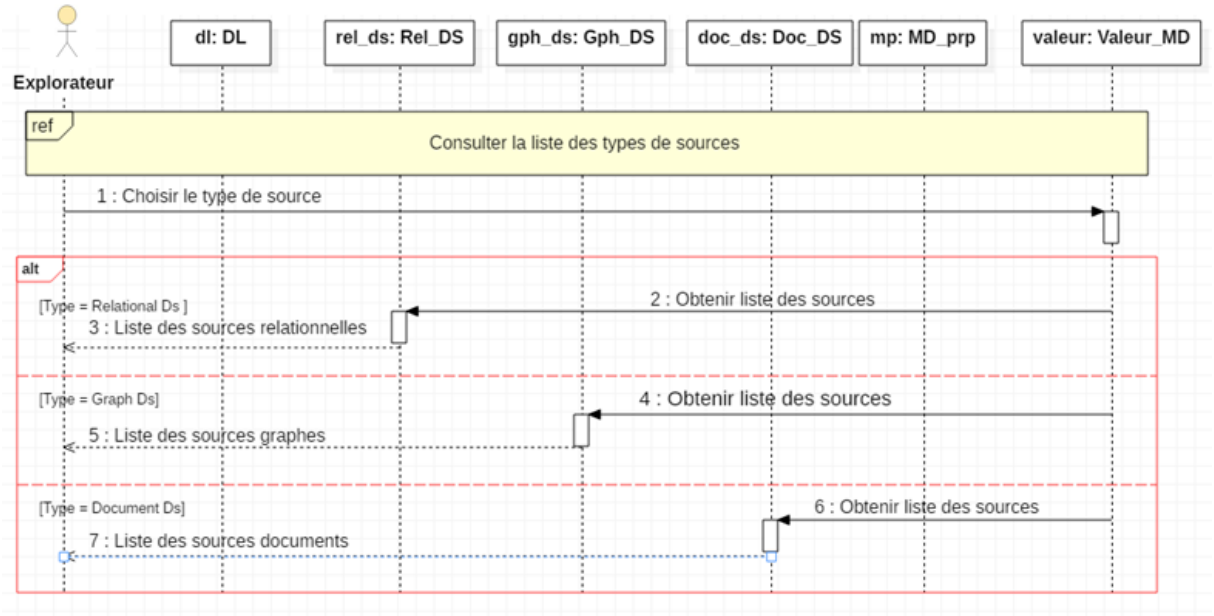


Figure 4.3 : Diagramme de séquence de cas d'utilisation « Consulter la liste des sources d'un type de source de données de DL ».

- **Cas d'utilisation** « Consulter la liste des tables relationnelles d'une source relationnelle de DL » (voir figure 4.4).

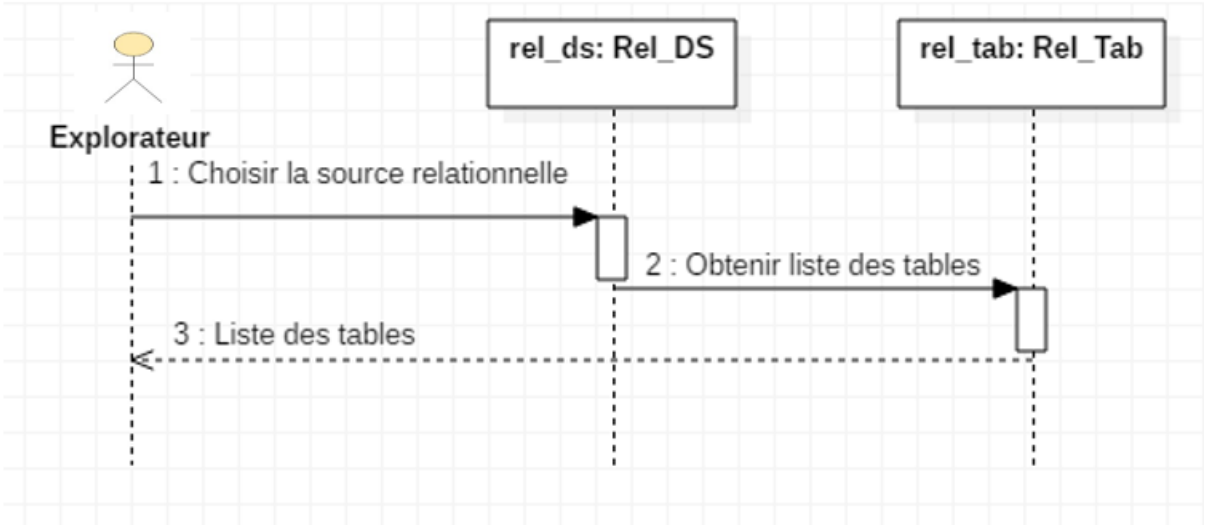


Figure 4.4 : Diagramme de séquence de cas d'utilisation « Consulter la liste des tables relationnelles d'une source relationnelle de DL »

- **Cas d'utilisation**« Consulter la liste des colonnes d'une table relationnelle » (voir figure 4.5).

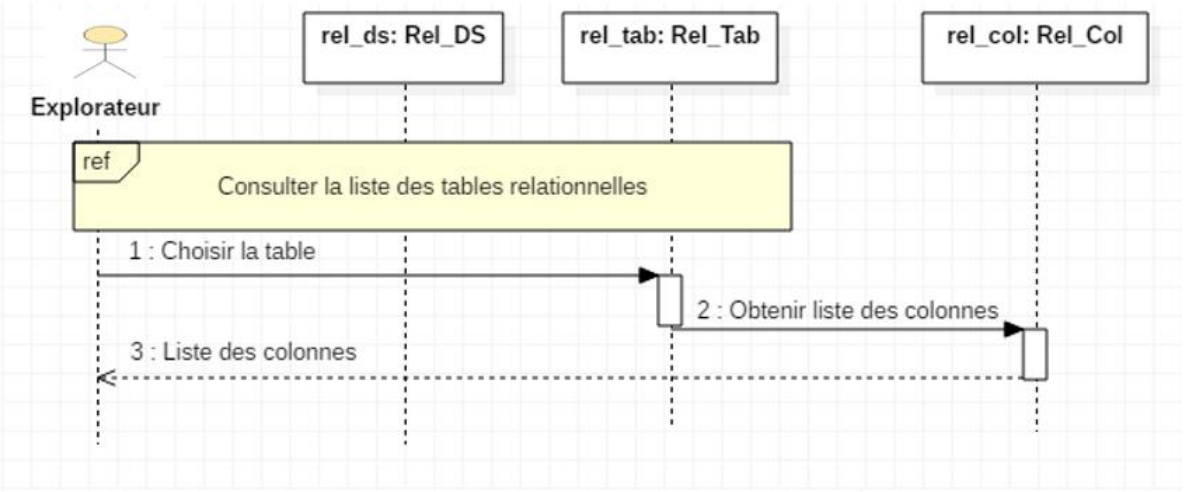


Figure 4.5 : Diagramme de séquence de cas d'utilisation « Consulter la liste des colonnes d'une table relationnelle ».

- **Cas d'utilisation** « Consulter la liste des références relationnelles d'une source Relationnelle » (voir figure 4.6).

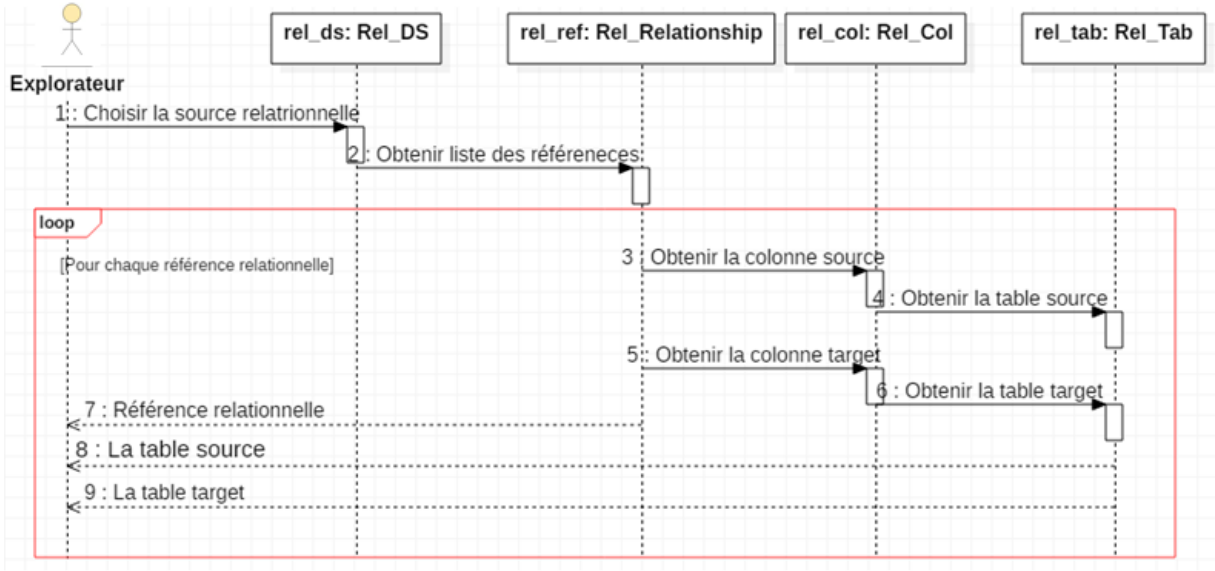


Figure 4.6 : Diagramme de séquence de cas d'utilisation « Consulter la liste des références relationnelles d'une source relationnelle ».

- **Cas d'utilisation «** Consulter la liste des collections d'une source document de DL » (voir figure 4.7).

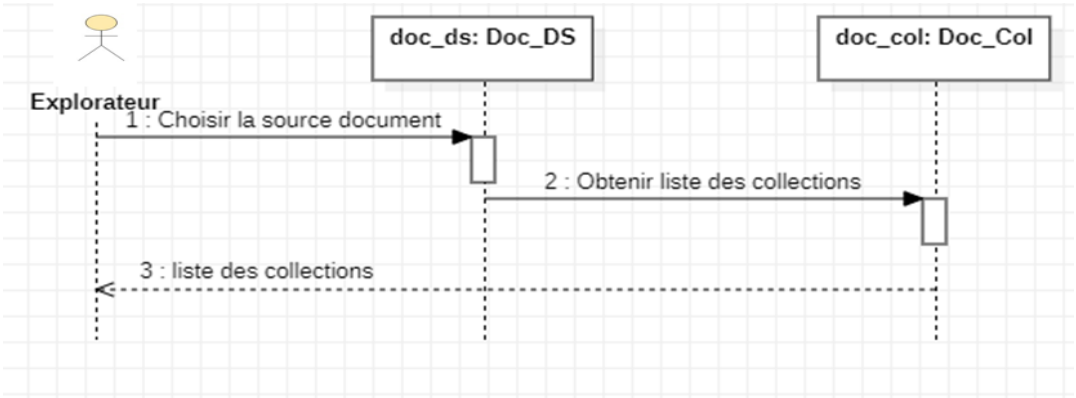


Figure 4.7 : Diagramme de séquence de cas d'utilisation **«** Consulter la liste des collections d'une source document de DL »

- **Cas d'utilisation** « Consulter la liste des items d'une collection document » (voir figure 4.8).

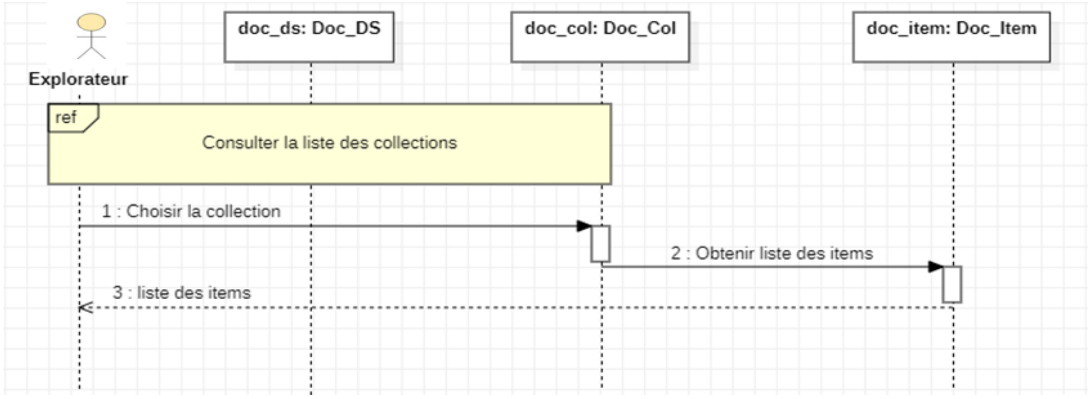


Figure 4.8: Diagramme de séquence de cas d'utilisation « Consulter la liste des items d'une collection document ».

- **Cas d'utilisation «** Consulter la liste des items composants d'un item particulier » (voir figure 4.9).

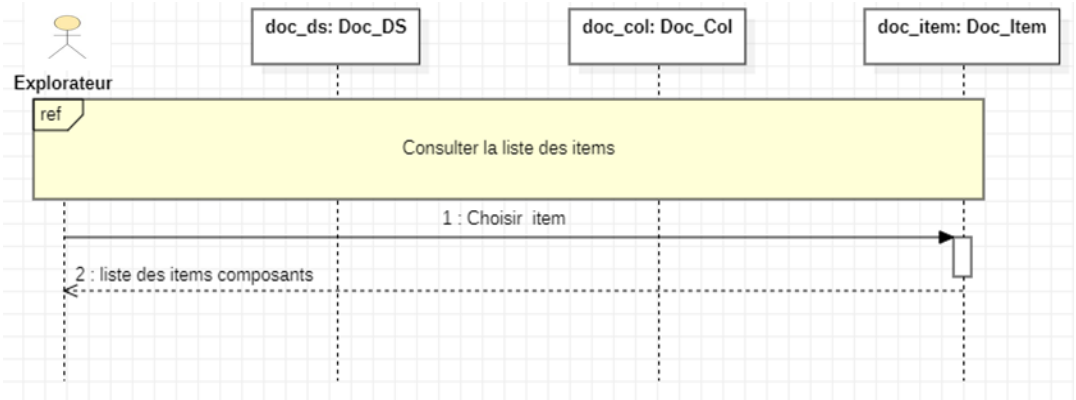


Figure 4.9 : Diagramme de séquence de cas d'utilisation « Consulter la liste des items composants d'un item particulier ».

- **Cas d'utilisation** « Consulter La liste des références documents d'une source document » (voir figure 4.10).

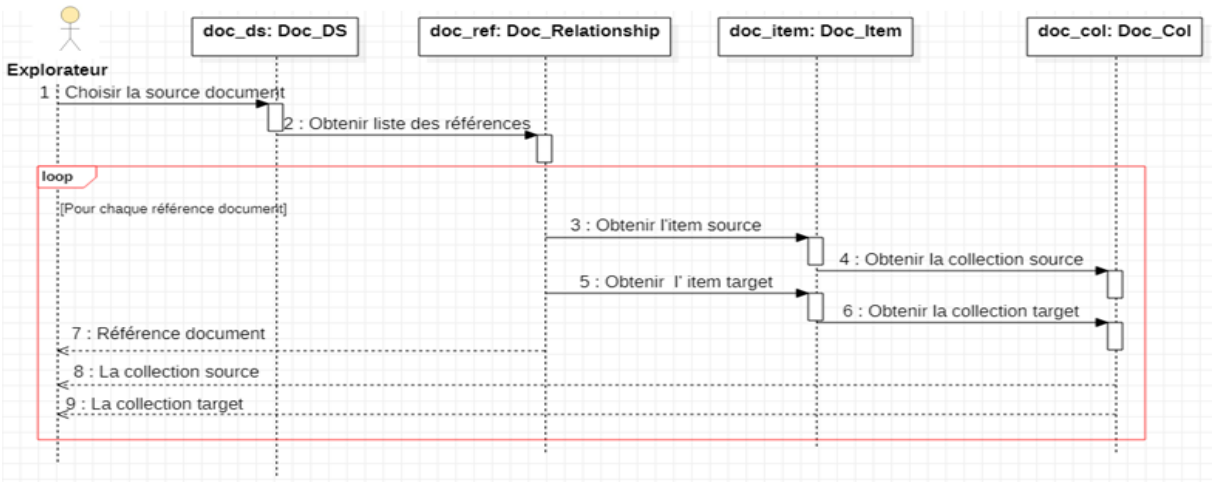


Figure 4.10 : Diagramme de séquence de cas d'utilisation « Consulter La liste des références documents d'une source document ».

- **Cas d'utilisation** « Consulter la liste des vertices ou edges d'une source graphe » (voir figure 4.11).

  Scénario : Consulter la liste des vertices d'une source graphe.

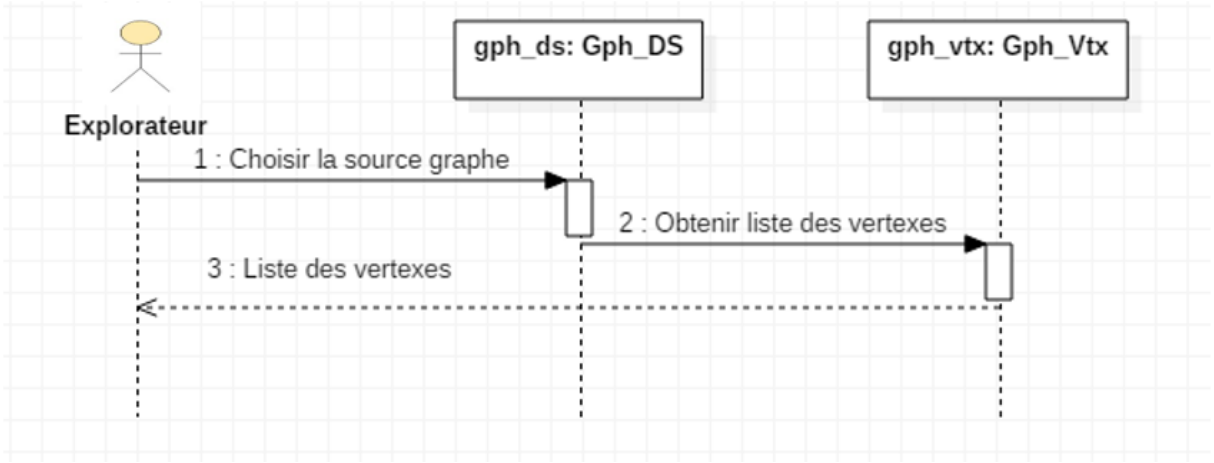


Figure 4.11 : Diagramme de séquence de cas d'utilisation « Consulter la liste des vertices ou edges d'une source graphe ».

Remarque : Le scénario des edges se fait de la même manière en ajoutant les edges, (on utilise les vertices dans ce scénario pour récupérer la source et la cible des edges).

- **Cas d'utilisation «** Consulter la liste des propriétés d'un vertex ou d'une edge » (voir figure 4.12).

  Scénario : Consulter la liste des propriétés d'un vertex.

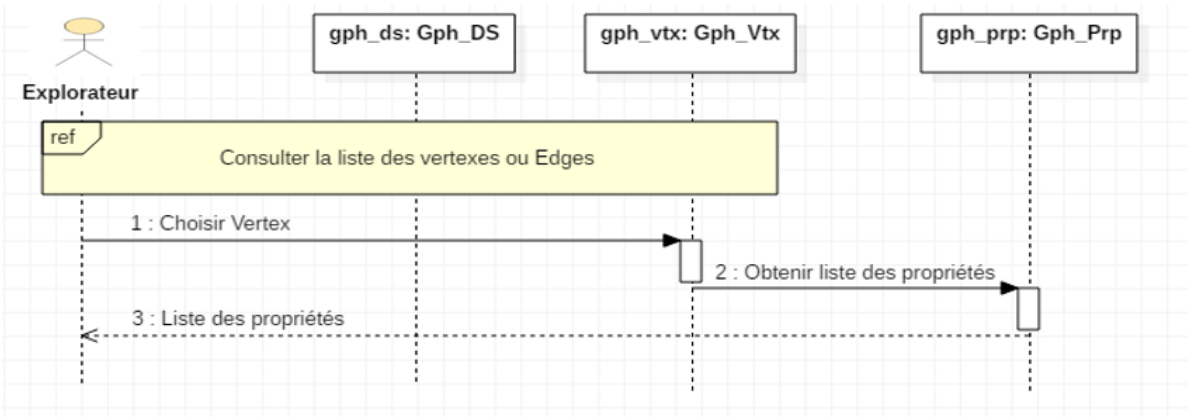


Figure 4.12 : Diagramme de séquence de cas d'utilisation « Consulter la liste des propriétés d'un vertex ou d'une edge ».

Remarque : Le scénario des edges se fait de la même manière en remplaçant les vertices par les edges.

- **Cas d'utilisation «** Accès aux données d'une source graphe » (voir figure 4.13).

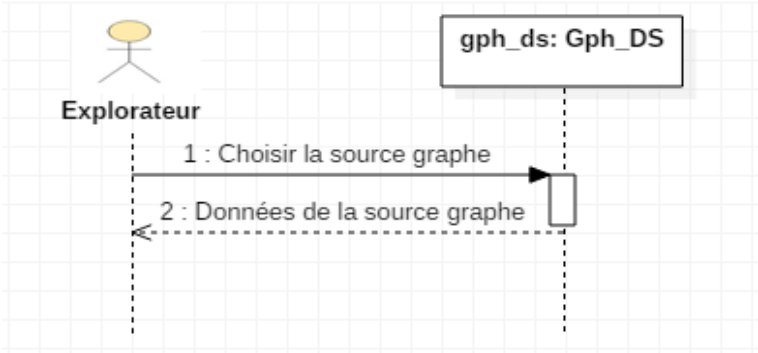


Figure 4.13 : Diagramme de séquence de cas d'utilisation « Accès aux données d'une source graphe »

- **Cas d'utilisation «** Accès aux données » (voir figure 4.14).

  Scénario : Accès aux données d'une table relationnelle.

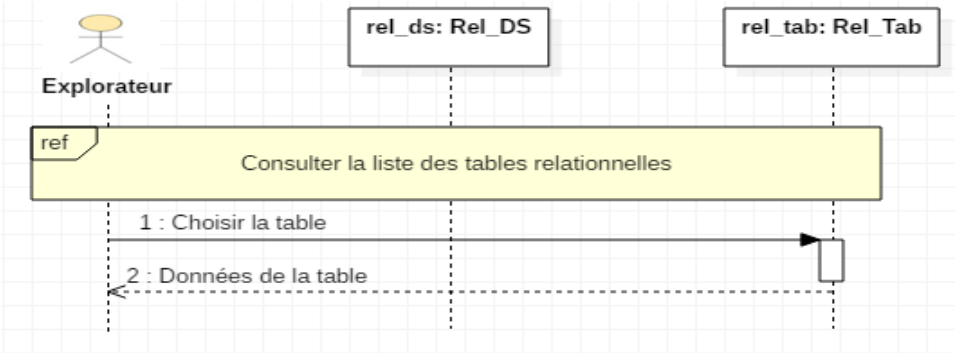


Figure 4.14 : Diagramme de séquence de cas d'utilisation « Accès aux données ».

- **Cas d'utilisation «** Accès aux métadonnées » (voir figure 4.15).

  Scénario : Accès aux métadonnées d'une table relationnelle.

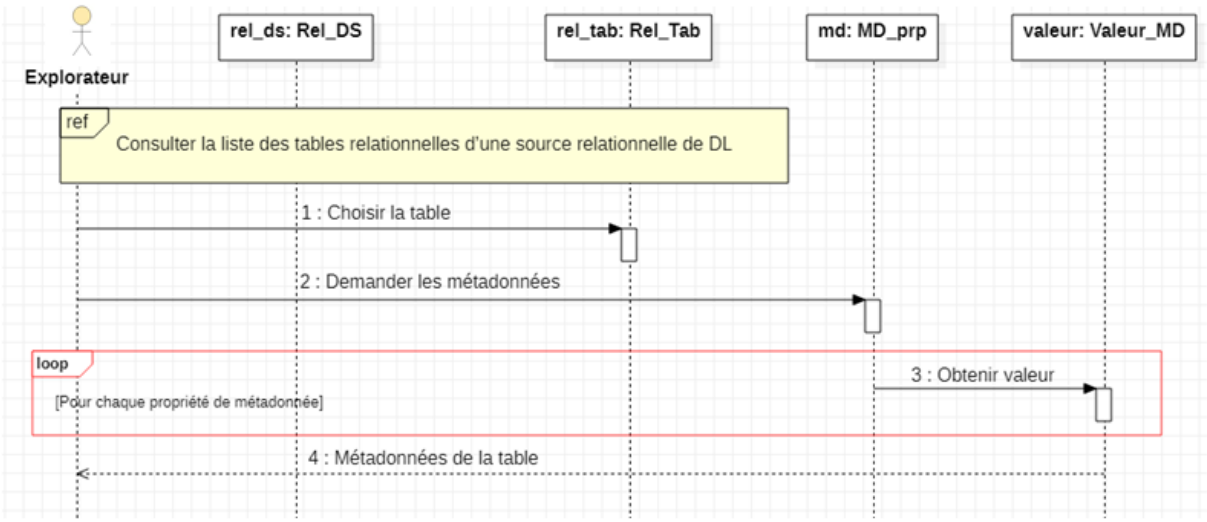


Figure 4.15 : Diagramme de séquence de cas d'utilisation « Accès aux métadonnées ».

- **Cas d'utilisation «** Consulter la liste des entités ou liens sémantiques (avec toutes les sources de DL) » (voir figure 4.16).

Scénario : Consulter la liste des entités sémantiques.

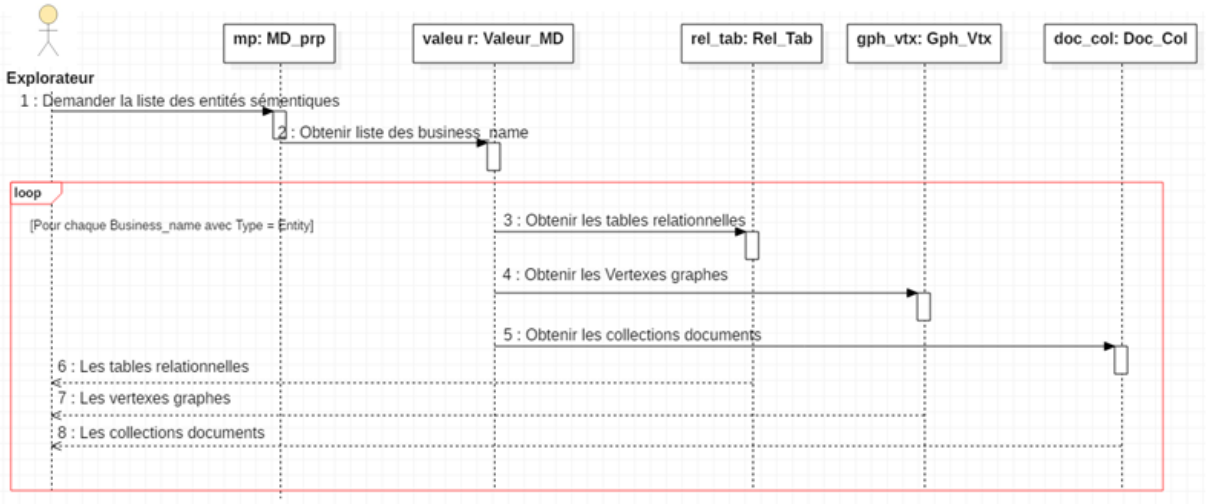


Figure 4.16 : Diagramme de séquence de cas d'utilisation « Consulter la liste des entités ou liens sémantiques (Avec toutes les sources de DL) ».

Remarque : Le scénario des liens sémantiques se fait de la même manière en remplaçant les vertices par les edges, les tables relationnelles par les références relationnelles, et les collections documents par les références documents.

- **Cas d'utilisation «** Rechercher un élément de DL » (voir figure 4.17).

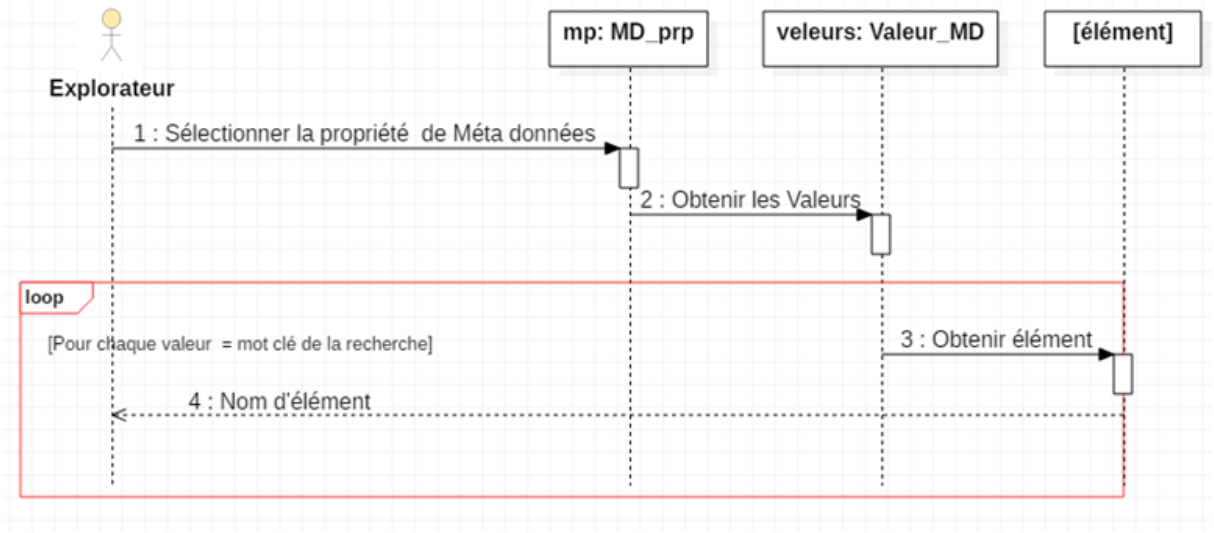


Figure 4.17 : Diagramme de séquence de cas d'utilisation « Rechercher un élément de DL ».

Remarque : Ce cas d'utilisation (Rechercher un élément de DL) peut être suivi par les autres cas par extension, par exemple quand on trouve un élément, on peut afficher ses données, ses Métadonnées ...etc.

4.3 Conception

La phase de conception permet d'ébaucher et de détailler la solution proposée. Les principales étapes sont la conception du déploiement, le développement du modèle logiciel d'exploitation du système, et la conception du modèle logique.

4.3.1 Concevoir le déploiement

Le modèle de déploiement considère chaque nœud comme un poste de travail. Il exprime la répartition physique des fonctions métier du système et permet de justifier la localisation des bases de données et des environnements de travail. Le modèle de déploiement aide à préciser la qualification des postes client, des réseaux et de leur sécurité physique, par rapport à des critères fonctionnels [45], Dans notre système nous avons un seul post de travail : Explorateu**r** qui permet l'exploration des lacs de données. Ce poste de travail est lié directement à un serveur de base de données qui permet de stocker les métadonnées du lac de données. La figure 4.18 représente le diagramme de déploiement de notre système.

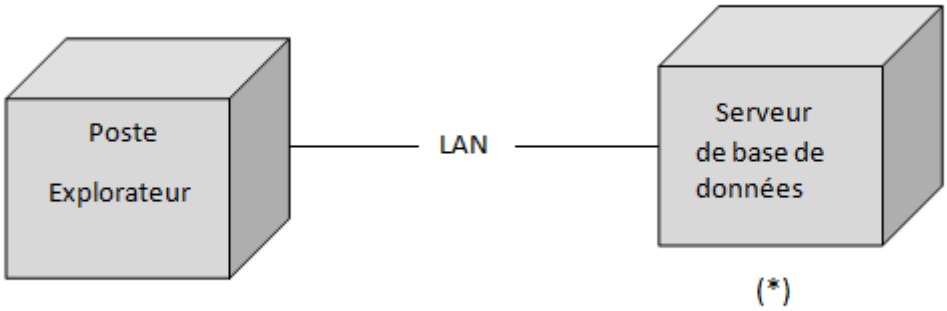


Figure 4.18: Diagramme de déploiement.

4.3.2 Concevoir le modèle d'exploitation

Un modèle d'exploitation et de composants permet de spécifier en général les composants et leurs répartitions sur les postes de travail, tels que les applications installées sur les postes de travail, les composants métiers déployés sur les serveurs, et les instances de bases de données implantées sur les serveurs également [45].

4.3.2.1 Découpage du système en applications

La première étape du développement d'un modèle d'exploitation est le découpage du système en applications. Le point de départ de cette étape est le regroupement des fonctions effectuées durant l'étape de capture des besoins fonctionnels en tant que cas d'utilisation, tout en respectant la définition des postes de travail [45]. La figure 4.19 montre la répartition des applications sur les postes de travail de notre système.

Figure 4.19 : Répartition des applications sur les postes de travail de notre système.

Remarquons que nous avons un seul poste de travail « Explorateur » avec une seule application « Exploration du Data Lake », vu que tous les cas d'utilisation de notre système concernent le même acteur et recourent à la même application.

(*) Dans le diagramme de déploiement et le modèle d'exploitation signifie qu'on peut avoir plusieurs instances.

4.3.2.2 Identification des composants distribués

La première façon d'identifier les composants distribués consiste à recenser les catégories d'analyse partagées par plusieurs applications, le critère de partage et de réutilisation n'est a posteriori pas suffisant : les composants distribués offrent en effet un découplage logiciel entre les applications et le métier. Ce découplage facilite la réutilisation, mais également la maintenance et l'évolution du système global [45]. Dans notre système aucun composant distribué entre plusieurs applications n'est identifié en raison du fait que nous avons identifié une seule application qui est installée dans un seul poste de travail.

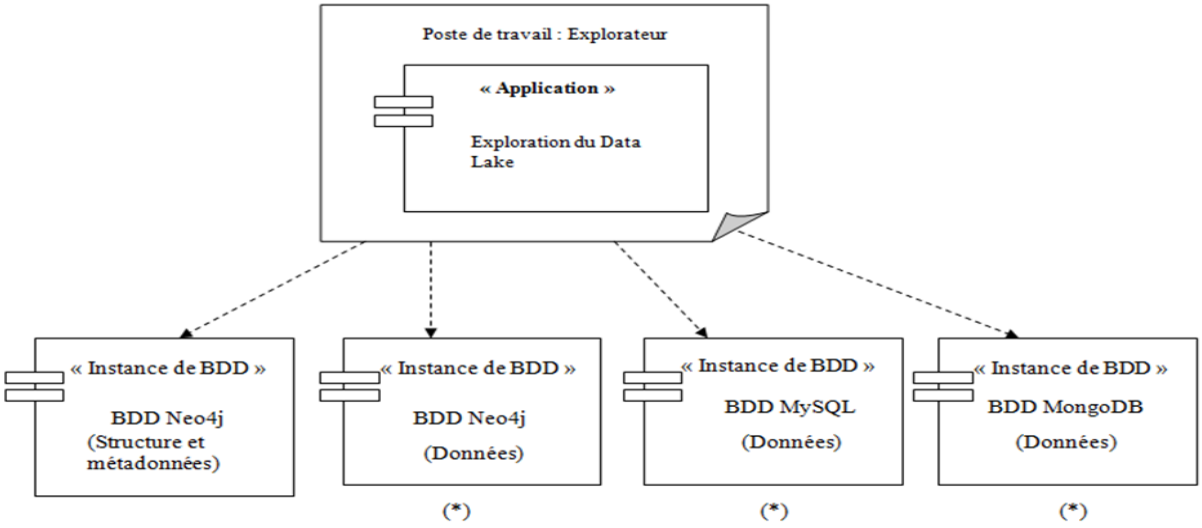


Figure 4.20 : Modèle d'exploitation de notre système.

4.3.2.3 Enumération des interfaces utilisateurs (IHM) des applications

Si les composants communiquent par le biais de leurs interfaces, les applications, quant à elles, sont utilisables par le biais de leurs interfaces utilisateur ou IHM (Interface Homme Machine). L'identification des interfaces Homme-machine sert à compléter l'architecture de l'exploitation en identifiant des composants IHM transverses aux applications, augmentant ainsi la réutilisation dans la conception de la couche de présentation [45]. Le tableau 4.1 illustre l'ensemble des Vues IHM de notre système :

	Vue IHM	**Description**
V1	Affichage d'une grille des types de source.	Visualisation d'une liste des types de sources d'un DL sélectionné.
V2	Affichage d'une grille des sources de données d'un DL sélectionné.	Visualisation d'une liste des sources de données d'un type de sources sélectionné, à partir d'une liste filtrée des types de sources. (Extension de V1)

V3	Affichage d'une grille des tables relationnelles d'une source relationnelle sélectionnée.	Visualisation d'une liste des tables relationnelles d'une source relationnelle sélectionnée, à partir d'une liste filtrée des sources relationnelles. (Extension de V2)
V4	Affichage d'une grille des colonnes d'une table relationnelle sélectionnée.	Visualisation d'une liste des colonnes d'une table relationnelle sélectionnée, à partir d'une liste filtrée des tables relationnelles. (Extension de V3)
V5	Affichage d'une grille des références relationnelles d'une source relationnelle sélectionnée.	Visualisation d'une liste des références relationnelles d'une source relationnelle sélectionnée, à partir d'une liste filtrée des sources relationnelles. (Extension de V2)
V6	Affichage d'une grille des collections d'une source document sélectionnée.	Visualisation d'une liste des collections d'une source document sélectionnée, à partir d'une liste filtrée des sources documents. (Extension de V2)
V7	Affichage d'une grille des items d'une collection document sélectionnée.	Visualisation d'une liste des items d'une collection document sélectionnée, à partir d'une liste filtrée des collections documents. (Extension de V6)
V8	Affichage d'une grille des items composants d'un item sélectionné.	Visualisation d'une liste des items composant d'un item sélectionnée d'une manière récursive, à partir d'une liste filtrée des items. (Extension de V7)
V9	Affichage d'une grille des références documents d'une source document sélectionnée.	Visualisation d'une liste des références documents d'une source document sélectionnée, à partir une liste filtrée des sources document. (Extension de V2)
V10	Affichage d'un méta graphe (sous graphe) des vertices et edges d'une source graphe sélectionnée.	Visualisation de l'ensemble des vertices ou edges d'une source graphe sélectionnée, à partir une liste filtrée des sources graphe sous forme des sous graphes. (Extension de V2)
V11	Affichage d'une grille des propriétés d'un vertex ou edge graphe sélectionné.	Visualisation d'une liste des propriétés d'un vertex ou edge graphe sélectionné, à partir d'une liste filtrée des vertices ou edges. (Extension de V10)

V12	Affichage d'un tableau hiérarchisé des entités et liens sémantiques.		Visualisation d'une liste des entités et liens sémantiques de data Lake.
V13	Accès aux données	Affichage d'une grille (Source relationnelle).	- Visualisation des données d'une table relationnelle sélectionnée. (Extension de V3) - Visualisation des données d'une colonne (s) sélectionnée. (Extension de V4) - Visualisation des données d'une référence relationnelle sélectionnée. (Extension de V5)
		Affichage d'un graphe (source graphe).	- Visualisation des données d'une source graphe sélectionnée. (Extension de V2) - Visualisation des données d'un vertex ou d'un edge graphe sélectionné. (Extension de V10) - Visualisation des données d'une propriété (s) sélectionnée. (Extension de V11)
		Affichage d'un document (source document).	- Visualisation des données d'une collection document sélectionnée. (Extension de V6) - Visualisation des données d'un item(s) sélectionné (Extension de V7)
V14	Affichage d'un tableau des métadonnées.		Visualisation d'un tableau des métadonnées de n'importe quel élément de Dl. (Extension de tous les autres vues IHM)
V15	Affichage d'une grille des résultats de la recherche sur un élément de DL.		Visualisation d'une liste des noms et types des éléments correspondent à un mot clé entré.

Tableau 4.1 : Vues IHM de notre système.

4.3.3 Concevoir le modèle logique

Pour développer le modèle logique de notre système, nous avons utilisé le modèle graphe au lieu d'un modèle relationnel en raison de la flexibilité des graphes et leur gestion des liens sans clés primaires et étrangères.

Pour traduire notre modèle objet (Diagramme de classe) en un modèle logique (graphe), nous appliquons les règles de passage indiqués ci-dessous :

Diagramme de classe	**Graphe**
La classe «**DL**»	Plusieurs nœuds possédant le label « DL » et différents noms.
La classe « **Gph_DS**»	Plusieurs nœuds possédant le label « Gph_DS»et différents noms.
La classe «**Doc_DS**»	Plusieurs nœuds possédant le label « Doc_DS» et différents noms.
La classe «**Rel_DS**»	Plusieurs nœuds possédant le label « Rel_DS» et différents noms.
La classe «**Gph_Vtx**»	Plusieurs nœuds possédant le label « Gph_Vtx» et différents noms.
La classe «**Gph_Edge**»	Plusieurs nœuds possédant le label « Gph_Edge» et différents noms.
La classe «**Gph_prp**»	Plusieurs nœuds possédant le label « Gph_Prp» et différents noms.
La classe «**Doc_Col**»	Plusieurs nœuds possédants le label « Doc_col» et différents noms.
La classe «**Doc_Item**»	Plusieurs nœuds possédants le label « Doc_item» et différents noms.
La classe «**Doc_Relationship**»	Plusieurs nœuds possédants le label «Doc_Relationship» et différents noms.
La classe «**Rel_Tab**»	Plusieurs nœuds possédants le label « Rel_Tab» et différents noms.
La classe «**Rel_Col**»	Plusieurs nœuds possédants le label « Rel_Col» et différents noms.
La classe «**Rel_Relationship**»	Plusieurs nœuds possédants le label « Rel_Relationship» et différents noms.
La classe «**MD_prp**»	Plusieurs nœuds possédants le label «MD_Prp» et différents noms.
La classe d'association «**Valeur_md**» qui est entre la classe Md_prpet les autres classes	Des relations de type «Has_MD_Prp» entre les nœuds avec label «MD_Prp » et tous les nœuds des autres labels avec des valeurs différentes de la propriété de lien avec le nom «valeur ».

Association de type composition /Agrégation	Des relations de type «ComposedOf» entre les nœuds concernés.
Association «**has_primary_key**» Entre la classe Doc_Col et Doc_Item , Et entre la classe Rel_Tab et Rel_Col	Des relations de type «has_primary_key » qui lient respectivement chaque nœud avec le label « Rel_Tab » \«Doc_Col » avec un ou plusieurs nœud avec label «Rel_Col»\ Doc_Item».

Tableau 4.2 : Règles de passage de modèle objet au modèle graphe.

4.5 Conclusion

Ce chapitre a été consacré à l'analyse et la conception du notre système. Dans l'étape d'analyse, nous avons présenté le diagramme de classe global, ainsi que l'ensemble des diagrammes de séquences qui traitent les cas d'utilisation identifiés dans la capture des besoins fonctionnels. Ensuite, dans l'étape de conception, nous avons présenté le modèle de déploiement et d'exploitation, l'ensemble des vues hommes-machines et enfin le modèle logique de notre système en citant l'ensemble des règles de passage de modèle de classes d'analyse au modèle graphe.

Chapitre 05 : Mise en œuvre

5.1 Introduction

Dans ce chapitre, nous allons présenter la phase d'implémentation de notre système d'exploration du lac de données qui traite l'aspect pratique de notre projet de fin d'étude. Nous allons d'abord commencer par une brève description des environnements de travail, les technologies et l'ensemble des logiciels que nous avons utilisés dans la réalisation de l'application ainsi que le jeu de données utilisé pour implémenter un lac de données et montrer l'accès à ses données. Ensuite, nous allons décrire et présenter un aperçu des interfaces les plus importantes de notre application.

5.2 Choix techniques

Les outils et logiciels utilisés dans la phase d'implémentation de notre application sont les suivants.

5.2.1 Environnement de développement

5.2.1.1 Java

Java est un langage de programmation orienté-objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au *SunWorld*. La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java [46].

Java est une technologie incroyablement polyvalente et convient parfaitement aux implémentations complexes de logique d'entreprise. Elle est également utilisée dans des applications IoT. Aujourd'hui, ce langage de programmation est présent sur des milliards d'appareils mobiles dans le monde entier car un grand nombre d'applications sont écrites en Java. De plus, si quelqu'un s'intéresse aux Big Data qui est le cœur et le point de départ de notre projet, il est nécessaire de connaître également ce langage de programmation, c'est pour cela que nous l'avons utilisé comme langage de codage de notre système [47].

Java est un langage puissant, populaire et connu comme le " roi de la programmation". Il présente des avantages tels que le multithreading, l'extensibilité, la gestion de la mémoire, la haute sécurité, le support communautaire…etc. qui le rendent extrêmement pertinent pour différents projets de développement d'applications [47].

5.2.1.2 JDK *(Java Development Kit*)

JDK est un kit de développement logiciel permettant de développer des applications en Java et désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être compilé, transformé en byte code destiné à la machine virtuelle Java.

Les principaux composants du JDK sont une sélection d'outils de programmation, incluant : le chargeur d'application, le compilateur, l'archiveur, le désassembleur, etc. En plus de ces outils, le JDK contient le progiciel JRE (*java Runtime Environnement*) qui fournit des bibliothèques de classes Java [48].

5.2.1.3 NetBeans

NetBeans est un environnement de développement intégré gratuit et à code source ouvert destiné au développement d'applications sous Windows, Mac, Linux et Solaris.

NetBeans IDE propose des outils de première classe pour le développement d'application Web d'entreprise, de bureau et mobiles Java, il est systématiquement le premier environnement IDE à prendre en charge les dernières versions de JDK, Java EE [7] et Java FX[8]. Il fournit des aperçus intelligents pour vous aider à comprendre et à gérer vos applications, y compris une prise en charge complète des technologies populaires telles que Maven.

Avec ses fonctionnalités de développement d'application de bout en bout, l'amélioration constante de Java Editor et ses améliorations en continu en termes de performances et de vitesse, NetBeans IDE établit la norme en matière de développement d'applications avec des technologies de pointe prêtes à l'emploi [49].

5.2.2 Environnement de stockage

5.2.2.1 MySQL

MySQL est un système de gestion de base de données relationnelle (SGBDR) open source qui appuie sur le langage de requête structuré SQL. Il est distribué sous double licence, une licence publique générale GNU et une propriétaire selon l'utilisation qui en est faite. La première version de MySQL est apparue en 1995 et l'outil est régulièrement entretenu.

Ce système est particulièrement connu des développeurs pour faire partie des célèbres quatuors **WAMP** (Windows, Apache, MySQL et PHP), **LAMP** (Linux) et **MAMP** (Mac). Ces packages sont si populaires et simples à mettre en œuvre que MySQL

[7] J2EE est une plate-forme fortement orientée serveur pour le développement et l'exécution d'applications distribuées.

[8] Java FX est une plate-forme logicielle pour créer et fournir des applications de bureau, ainsi que des applications Web riches pouvant s'exécuter sur une grande variété d'appareils.

est largement connu et exploité comme système de gestion de base de données pour des applications utilisant PHP. C'est d'ailleurs pour cette raison que la plupart des hébergeurs web proposent PHP et MySQL [50].

Voilà les caractéristiques principales de serveur MySQL [50]:

- MySQL est un serveur de base de données relationnelles SQL qui fonctionne sur de nombreux systèmes d'exploitation (dont Linux, Mac OS X, Windows, Solaris, FreeBSD…) et qui est accessible en écriture par de nombreux langages de programmation, incluant notamment PHP, Java, Ruby, C, C++, .NET, Python
- L'une des spécificités de MySQL c'est qu'il inclut plusieurs moteurs de bases de données et qu'il est par ailleurs possible au sein d'une même base de définir un moteur différent pour les tables qui composent la base.
- La réplication est possible avec MySQL et permet ainsi de répartir la charge sur plusieurs machines, d'optimiser les performances ou d'effectuer facilement des sauvegardes des données.

Nous avons géré les bases de données de serveur MySQL (version 5.7.24) de notre système en utilisant l'outil **Valentina studio** (version 12) comme outil client; la figure 5.1 représente l'environnement de travail de l'outil Valentina studio :

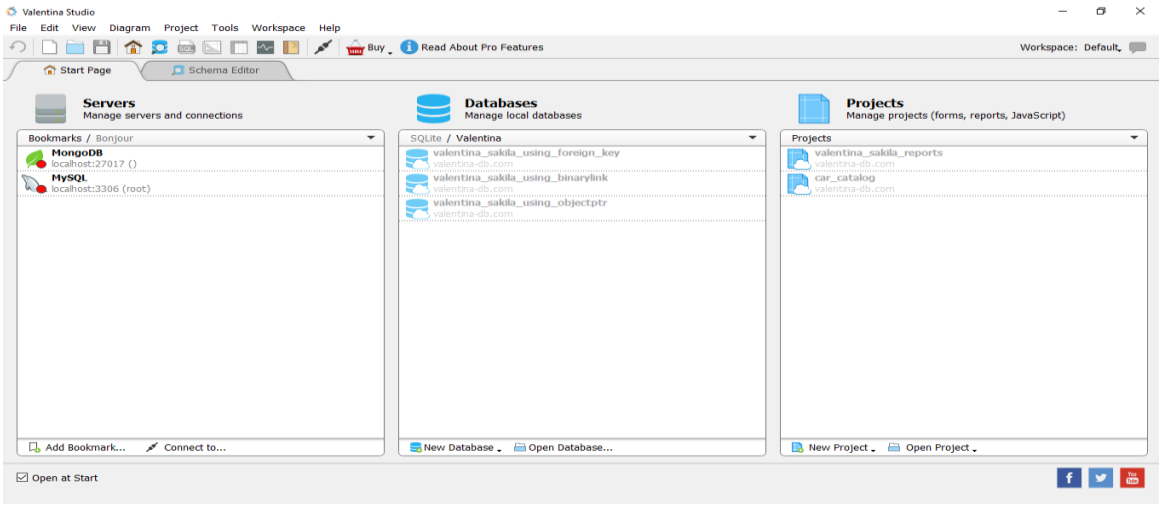


Figure 5.1: Valentina Studio.

5.2.2.2 MongoDB

MongoDB est un SGBD NoSQL orienté document écrit en C++ et développé par 10gen. Il se concentre sur la facilité d'utilisation, la performance et la grande évolutivité. MongoDB est disponible en 32 bits et 64 bits pour les environnements Windows et Unix. La façon par défaut de MongoDB de gérer les bases de données est le Shell JavaScript interactif qu'il fournit dès la sortie de la boîte. En outre, il existe des liaisons pour de nombreux langages de

programmation. Depuis de nombreux langages de programmation ont des types de données qui sont construits de paires clé-valeur certains soutiennent même JSON directement l'utilisation de ces langues est excellente pour créer des documents pour entrer dans MongoDB [51].

Voilà les caractéristiques principales de serveur MongoDB [52] :

- Chaque base de données MongoDB contient des collections, contenant elles-mêmes des documents. Chaque document peut être différent de l'autre et peut comporter un nombre de champs variables. La taille et le contenu de chaque document varient également.
- Le serveur MongoDB est considéré aujourd'hui comme un des systèmes maître esclave les plus performants qui offrent une sécurité de sauvegarde des données en cas de panne (structure des données dans les serveurs).
- MongoDB utilise le format JSON[9] mais en binaire (BSON[10]) et offre la possibilité de schémas dynamiques (GeoJSON[11] pour les données géospatiales). Cette utilisation offre un grand avantage car un bon nombre de langages de programmation utilise également ce format, ce qui offre un gain de temps conséquent.

Nous avons accédé aux bases de données de serveur MongoDB (version 5.0) de notre système en utilisant le client graphique **MongoDB Compass** (version 1.30.1) qui est un outil interactif pour interroger, optimiser et analyser les données MongoDB. La figure 5.2 représente l'environnement de travail de l'outil MongoDB Compass :

[9]**JSON** (*JavaScript Object Notation)* est un format standard utilisé pour représenter des données structurées de façon semblable aux objets Javascript.

[10]**BSON***(Binary JSON*) est un format d'échange de données informatiques utilisé principalement comme stockage de données et format de transfert de données par le réseau dans la base de données MongoDB.

[11]**GeoJSON** (*Geographic JSON*) est un format ouvert d'encodage d'ensemble de données géospatiales.

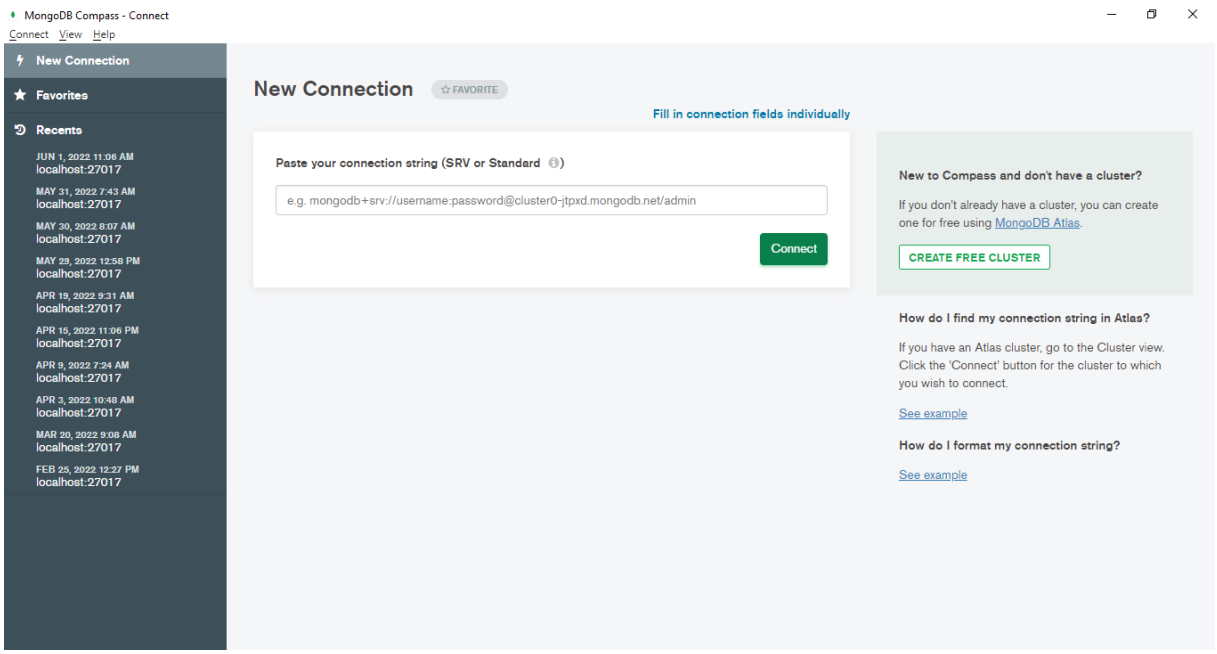


Figure 5.2 : MongoDB Compass.

5.2.2.3 Neo4j

Neo4j est un SGBD NoSQL orienté graphe écrite en Java. Développé par NeoTechnology. Il a vu le jour pour la première fois en 2007. Celle-ci est open-source et est distribué sous une double licence GPLv3 et AGPLv3 [25]. Les données d'une base de données neo4j sont stockées sur disque sous la forme d'une structure de données optimisée pour les réseaux de graphes. Le serveur neo4j peut être utilisé à la fois comme base embarquée sans aucun travail supplémentaire d'administration et comme un serveur à part entière, avec une interface REST[12] pour une intégration facilitée dans les environnements basés sur PHP, .NET ou JavaScript [53].

Voilà les caractéristiques principales de serveur Neo4j [53]:

- Un serveur avec une haute disponibilité via la sauvegarde en ligne et la réplication maître-esclave sont en bêta et les prochains sur la roadmap de livraison.
- Une structuration des données optionnelle voir absente, pour une facilité de changement du schéma et des migrations de données sans contraintes
- Modélisation facile de jeux de données de domaines normalement complexes

Nous avons utilisé la version **Community Edition** de serveur neo4j version 3.1.3 pour stocker, manipuler et accéder aux données graphes via le **neo4j browser**[13]. La Community Edition est une édition entièrement fonctionnelle de Neo4j, adaptée aux déploiements à une

[12] REST (*Representational State Transfer*) est un style architectural pour une interface de programme d'application (API) qui utilise les requêtes HTTP pour accéder aux données et les utiliser.

[13] Neo4j Browser est un Shell de commande (Cypher) qui nous permet d'interagir avec les bases de données graphes et de visualiser les informations qu'il contient

seule instance. Elle dispose d'un support complet pour les principales fonctionnalités Neo4j, la figure 5.3 représente l'interface Neo4j Browser.

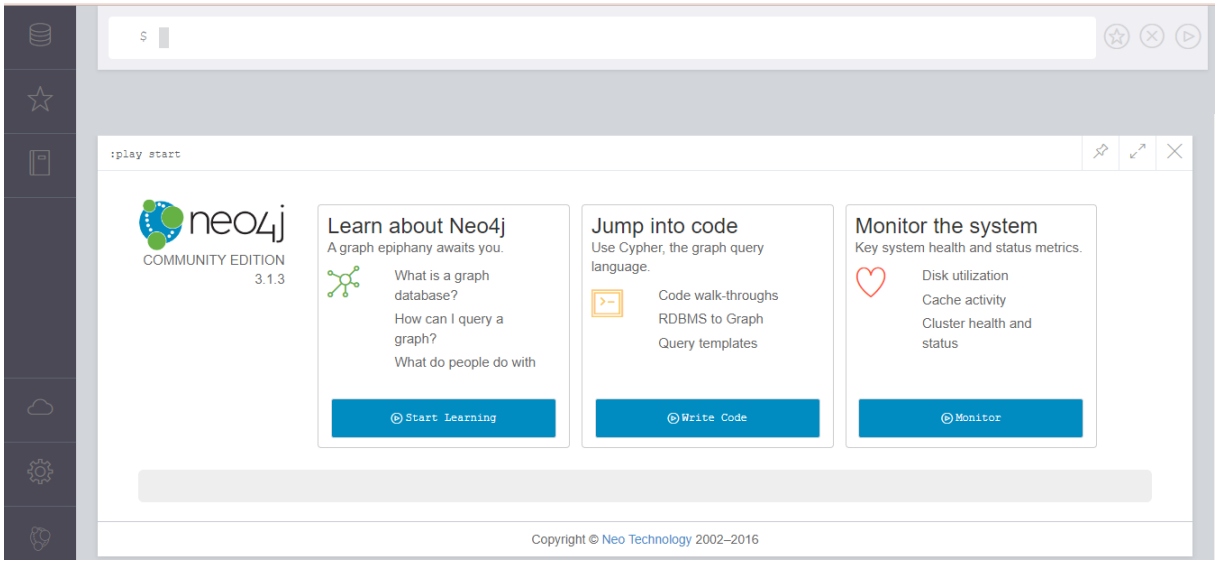


Figure 5.3 : Neo4j Browser.

5.3 Jeux de données

Dans le cadre de notre projet, nous avons mis en place un lac de données pour le réseau question / réponse StackExchange, qui se compose de trois sources de données NoSQL: une source relationnelle, une source graphe et une source document. Ce lac de données peut également être élargi à d'autres sources que les trois types mentionnés, ou à de nouveaux types de sources. Nous décrivons ci-dessous les trois sources actuelles de lac de données :

5.3.1 Source relationnelle : Pour stocker les données relationnelles, nous avons utilisé le serveur MySQL (voir la section 5.2.2.1). Notre base de données relationnelle contient six (06) tables relationnelles. Le nombre d'enregistrements de chaque table est présenté dans le tableau suivant :

Table	**Nombre d'enregistrements**
Flagtypes	4
Posts	726279
Posttypes	9
Users	669233
Votes	3595915
votetypes	15

Tableau 5.1 : Nombre d'enregistrements des tables de schéma relationnel.

– Pour alimenter notre base de données relationnelle, nous avons importé des données depuis des fichiers Csv téléchargés de la base de données StackExchange en utilisant

la fonctionnalité « Import from CSV… » de Valentina Studio, la figure 5.4 représente les résultats d'importation des données relationnelles dans la base de données relationnelle «data_import » dans Valentina Studio:

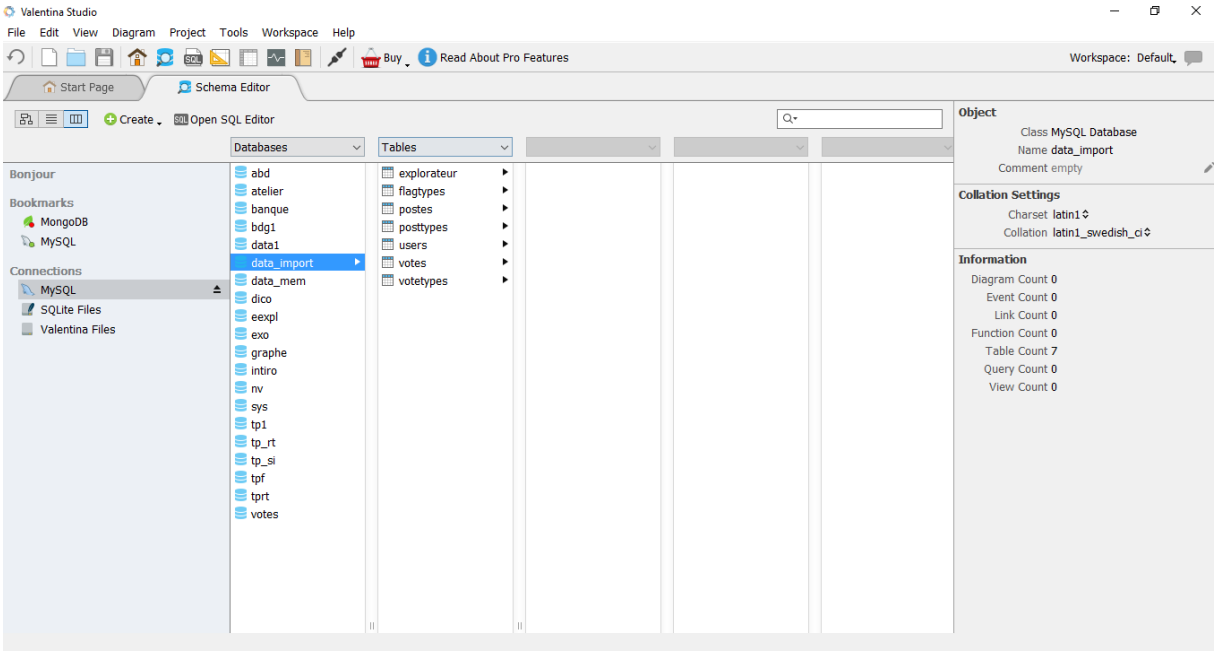


Figure 5.4 : Résultats d'importation des données relationnelles dans Valentina Studio.

5.3.2 Source document : Pour stocker les données de la source document, nous avons utilisé le serveur MongoDB (voir la section 5.2.2.2) avec le client MongoDB Compass. Notre base de données document contient quatre (04) collections, le nombre d'enregistrements de chaque collection est présenté dans le tableau suivant :

Collection	**Nombre d'enregistrements**
Badges	1125091
Posts	726279
Comments	2105709
Users	669233

Tableau 5.2 : Nombre d'enregistrements des collections de la source document.

– Pour alimenter notre base de données document, nous avons créé, dans un premier temps, les collections documents en utilisant la fonctionnalité « Create Collection » de MongoDB Compass; en second temps nous avons importé des données document StackExchange depuis des fichiers CSV en utilisant la fonctionnalité « add DATA », la figure 5.5 représente les résultats d'importation des données documents dans la base de données document « data_import » dans MongoDB Compass :

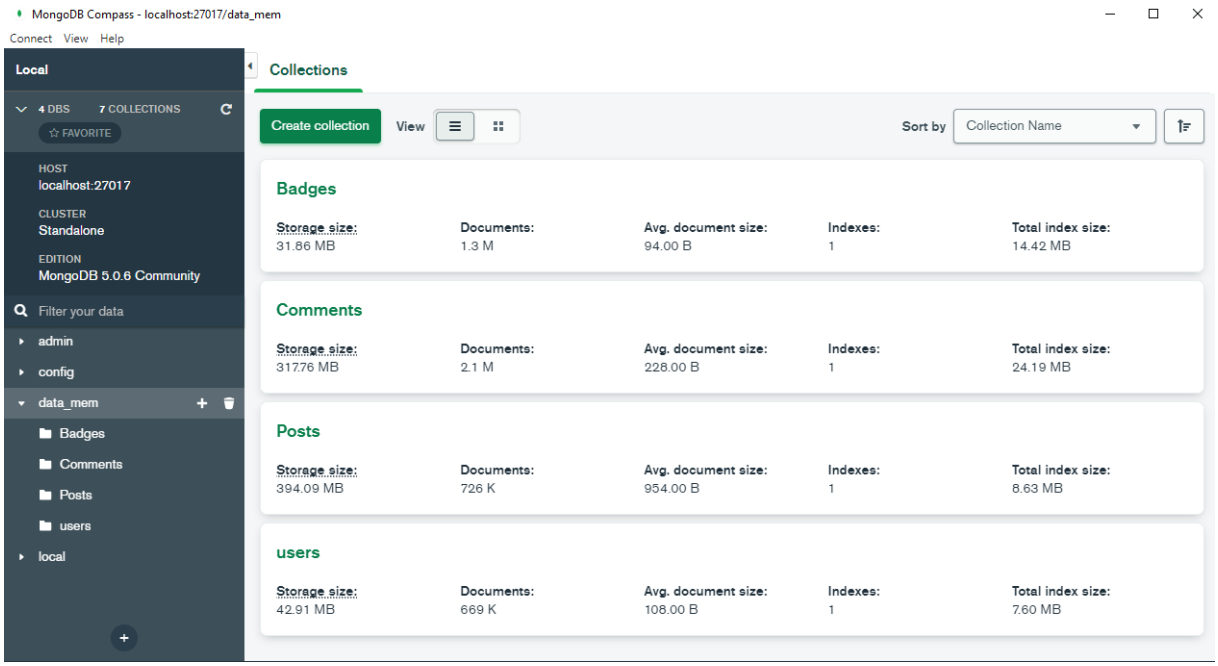


Figure 5.5 : Résultats d'importation des données documents dans MongoDB Compass.

5.3.3 Source graphe : Pour stocker les données de la source graphe, nous avons utilisé le serveur Neo4j (voir la section 5.2.2.3). Notre base de données graphe contient cinq (05) labels et neuf (09) Relationship types, le nombre de nœuds de chaque label et liens de chaque Relationship type est mentionné dans les deux tableaux suivants :

Label	**Nombre de nœuds**
Postlink	245152
Posts	726280
Comments	2105710
Users	669234
Tags	3063

Tableau 5.3 : Nombre de nœuds des labels label de la source graphe

Relationship type	**Nombre de liens**
ANSWER_OF	106580
COMMENT_ABOUT	2105709
HAS_COMMENTED	1996940
HAS_TAG	48922
LINK_FROM	170601
LINK_TO	241024

PARENT_OF	403906
POSTED_BY	208839
Synonym_OF	74

Tableau 5.4 : Nombre de Relationship des Relationship types de la source graphe.

– Pour alimenter notre base de données graphe, nous avons importé les données graphe de réseau StackExchange depuis des fichiers Csv, en utilisant des requêtes Cypher.

- Exemple de la requête pour importer les données de label **« Users »** :

```
:auto USING PERIODIC COMMIT
LOAD CSV FROM 'file:///users.csv' AS row
create(:Users {
id: toInteger(row[0]),
creationdate: row[1],
displayname: row[2],
accountid: toInteger(row[3])
})
```

- Exemple de la requête pour importer les données de Relationship type **« POSTED_BY »** :

```
:auto using Periodic Commit LOAD CSV with headers FROM "file:///Posts.csv" AS row FIELDTERMINATOR ',' match (n:Posts {id: toInteger(row.Id)}), (b:Users {id: toInteger(row.OwnerUserId)}) MERGE (n)-[:POSTED_BY ] - >(b)
```

La figure 5.6 représente les résultats d'importation des données graphe dans Neo4j :

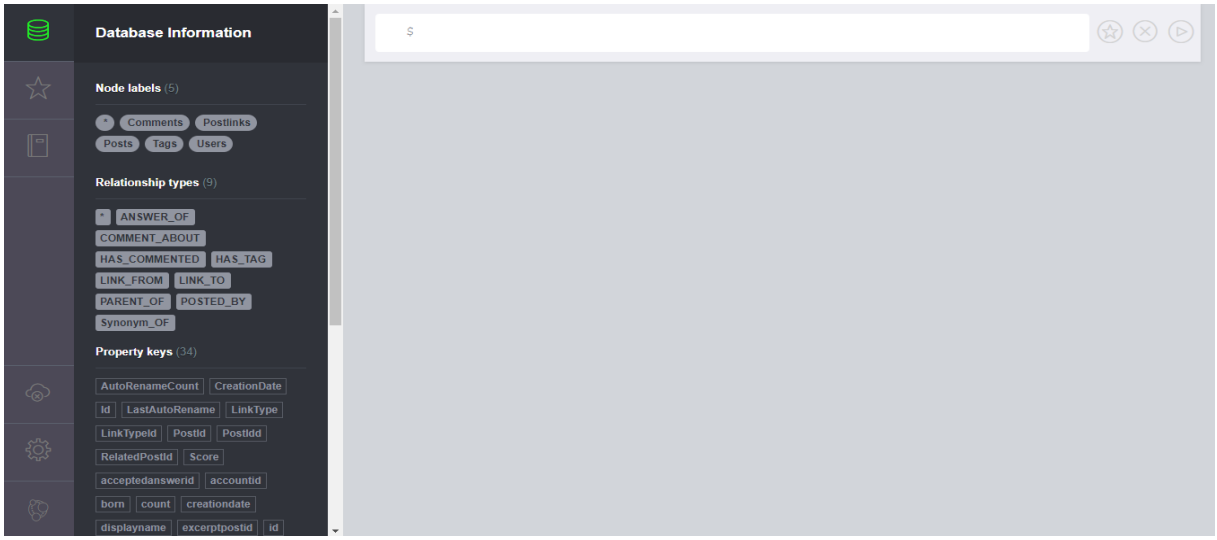


Figure 5.6 : Résultats d'importation des données graphe dans Neo4j.

5.4 Modèle et schéma de métadonnées et structure de lac de données StackExchange

Après la préparation du jeu de données des trois sources de lac de données StackExchange, nous avons créé un graphe qui représente le modèle de métadonnées pour stocker et manipuler les métadonnées et la structure de lac de données dans Neo4j. Cela signifie que nous avons une double utilisation de serveur Neo4j: pour stocker les données graphe et pour stocker et manipuler la structure et les métadonnées de lac de données.

Les deux tableaux suivants représentent les statistiques des Node Labels et Relationships types de graphe de métadonnées que nous avons créé :

Label	**Nombre de nœuds**
DL	1
Doc_DS	1
Rel_DS	1
Gph_DS	1
Rel_Tab	6
Rel_Relationship	5
Rel_col	37
Doc_col	4
Doc_Relationships	3
Doc_item	14
Gph_Vtx	5
Gph_Edge	9
Gph_prp	29
MD_Prp	4

Tableau 5.5 : Nombre de nœuds de schéma de métadonnées et structure.

Relationship types	**Nombre de liens**
ComposedOf	115
Has_MD_Prp	202
Has_Primary_key	15
Source	17
Target	17

Tableau 5.6 : Nombre de Relationships de schéma de métadonnées et structure.

Nous présentons ci-dessous (Figure 5.7 et Figure 5.8) le graphe de Métadonnées et de structure de lac de données que nous avons créé en le divisant par *Relationship types* pour des raisons de clarté.

La figure 5.7 représente les liaisons de type**«ComposedOf »**, **« source »**, **« target »** et **«Has_Primary_key ».**

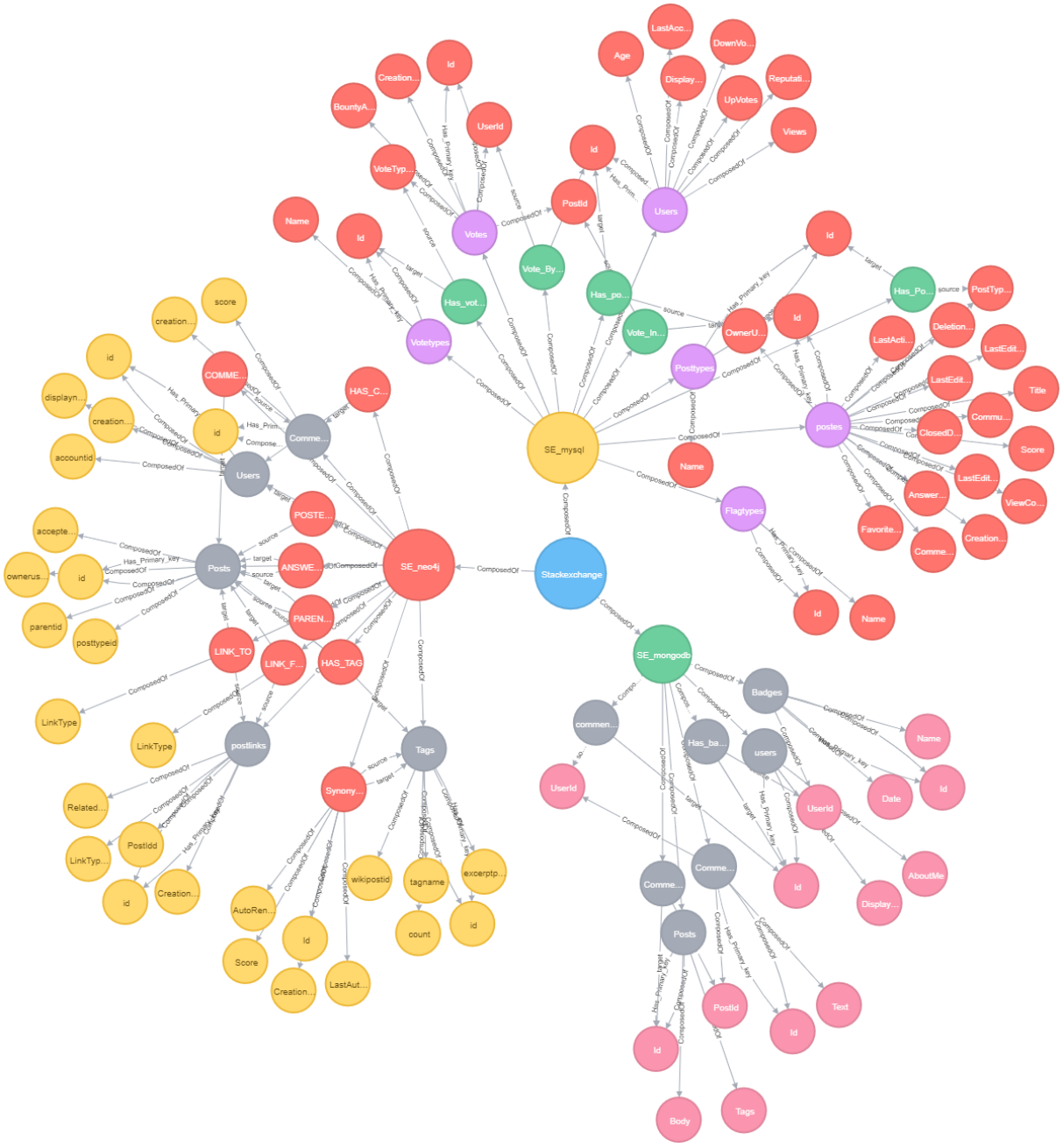


Figure 5.7 : Les liaisons ComposedOf, source, target et Has_Primary_key.

La figure 5.8 montre les liaisons de type **«Has_MD_prp »** en limitant le nombre de liaison à seulement 50 liaisons pour des raisons de lisibilité et de clarté, mais pratiquement la liaison Has_MD_prp existe avec tous les nœuds de lac de données.

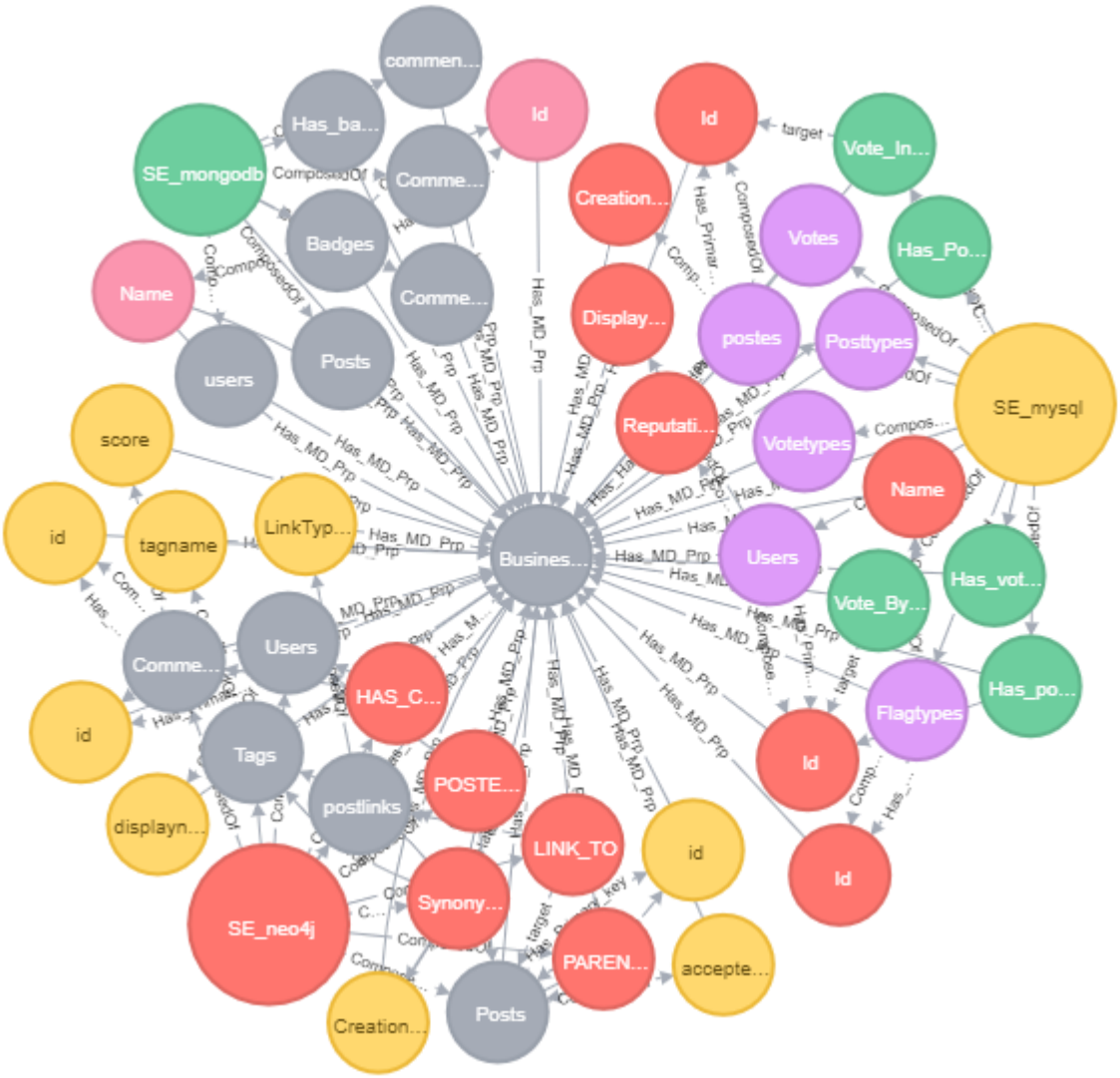


Figure 5.8 : Les liaisons Has_MD_prp.

5.5Application développée

5.5.1 Présentation de l'application

Nous avons développé une application d'exploration autour d'un lac de données basé sur des sources NoSQL (relationnelle, document et graphe), et extensible à d'autres types de sources.

Cette application se compose de quatre modules (comme illustré en figure 5.8): un module de structure et données du lac, un module de métadonnées, un module d'intégration sémantique des entités et liens et enfin, un module de recherche d'éléments dans le lac de données.

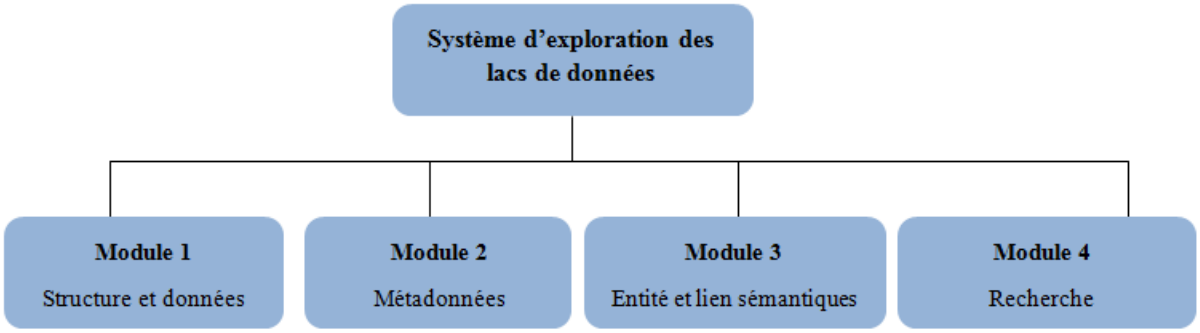


Figure 5.9 : Modules fonctionnelles de notre application.

5.5.2 Description des interfaces d'application

Dans cette section nous présentons une expérimentation de notre application, qui comprend la façon dont l'utilisateur peut accéder à l'application et les opérations qu'il peut effectuer.

– **Page d'accueil**

La figure 5.10 représente la page principale et l'interface d'accueil de notre application :

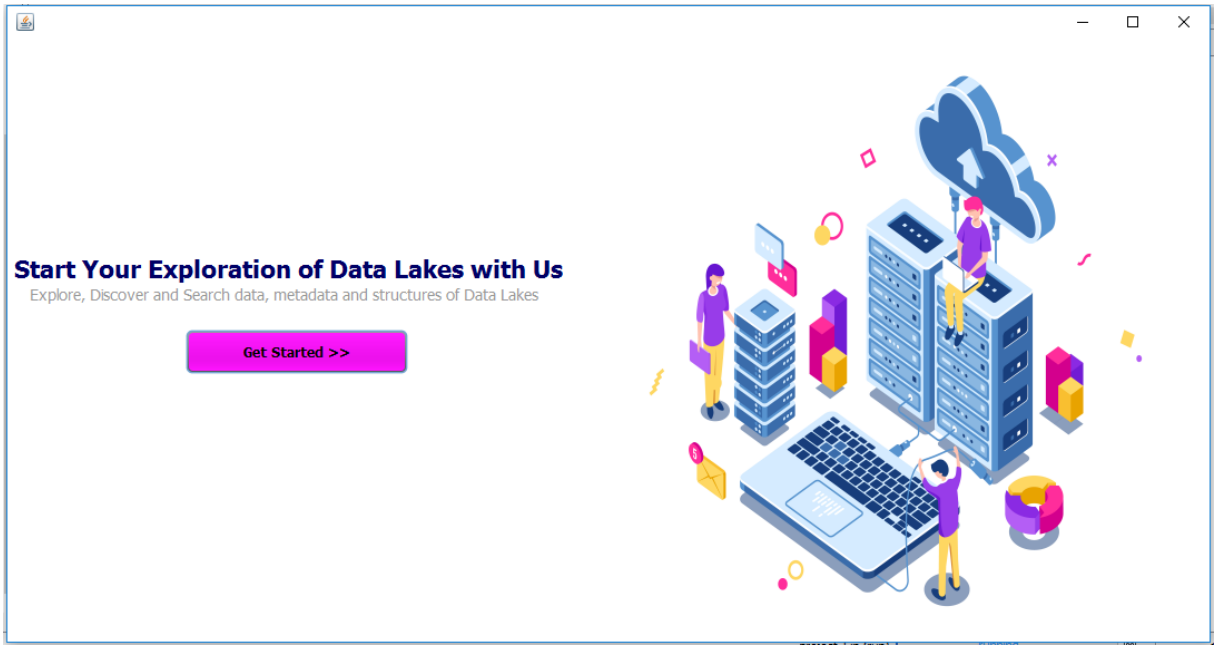


Figure 5.10 : Page d'accueil.

– **Page de connexion**

Pour commencer à utiliser le système, l'utilisateur doit s'authentifier avec son compte déjà créé. La figure 5.11 montre la page de connexion de notre application.

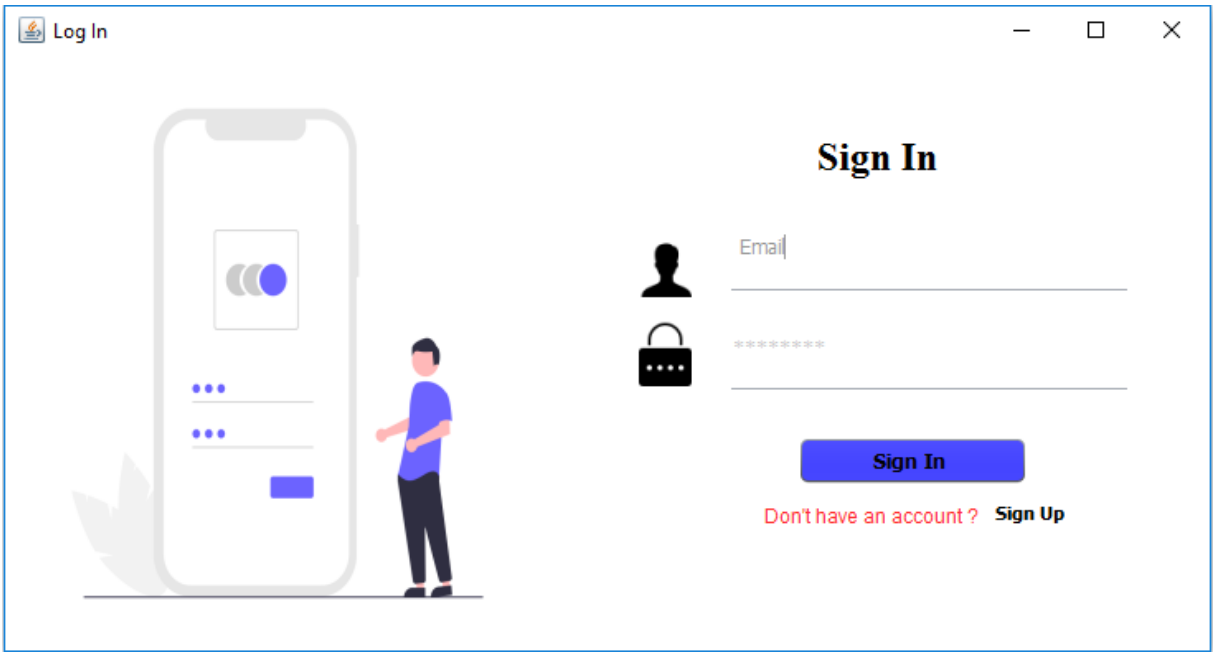


Figure 5.11 : Page de connexion.

– **Page d'inscription**

Si l'utilisateur n'a pas un compte, il doit en créer un avec son e-mail et son mot de passe. La figure 5.12 montre la page d'inscription de notre application :

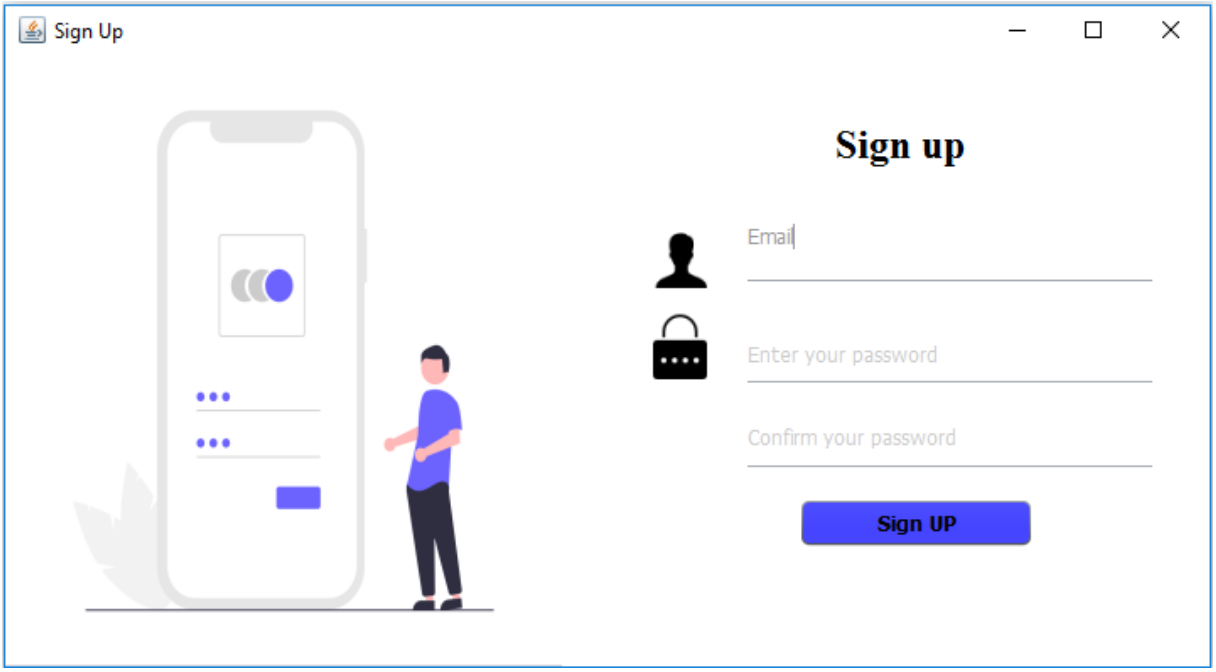


Figure 5.12 : Page d'inscription.

- L'utilisateur ne peut pas saisir de champs non valides, ou des informations incorrectes, car l'application contient un système de validation, la figure 5.13 représente des exemples d'erreurs que l'utilisateur pourrait obtenir lors de l'inscription ou la connexion.

Figure 5.13 : Exemple d'erreurs gérées par l'application.

– **Menu Principale**

Lorsque l'utilisateur s'authentifie, il sera dirigé vers la page suivante :

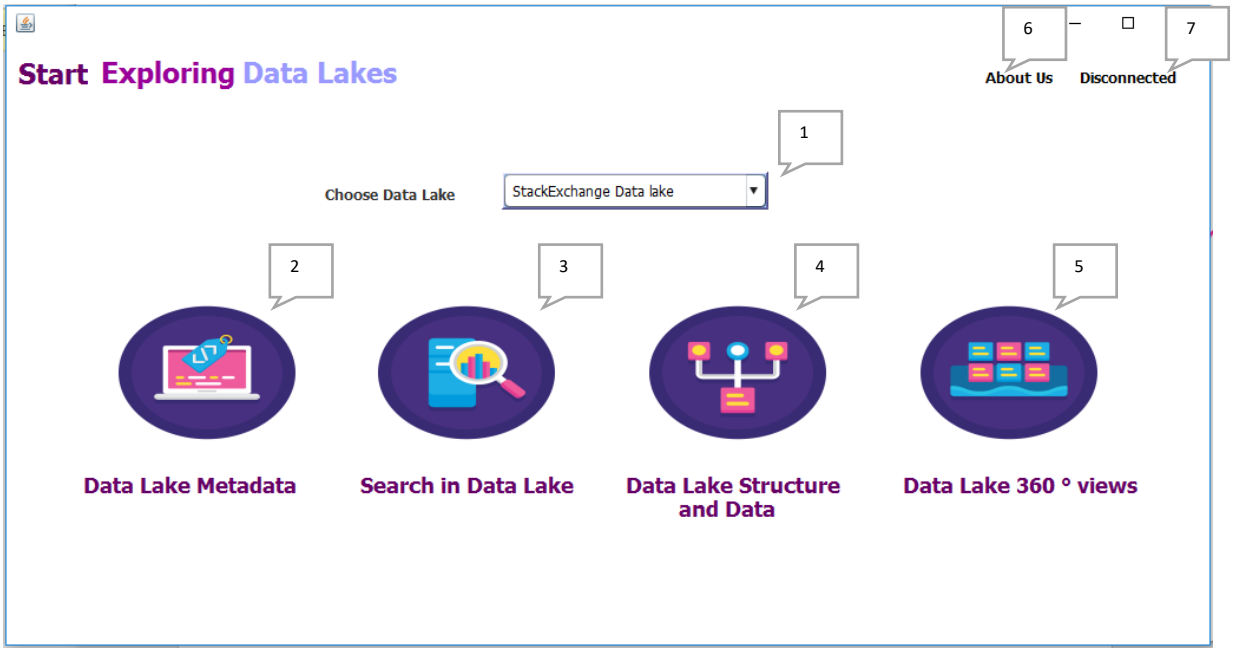


Figure 5.14 : Menu Principale.

Dans cette fenêtre l'utilisateur du système peut commencer à explorer le lac de données qu'il choisit dans la même fenêtre avec la liste déroulante **(1)** pour effectuer différentes actions, en cliquant sur l'un des buttons suivant :

- **(2)** Data lake Metadata.
- **(3)** Search in Data Lake.
- **(4)** Data Lake structure and data.
- **(5)** Data Lake 360° Views.

– **Data Lake Structure and Data**

L'utilisateur peut explorer la structure et les données de n'importe quelle source de données existant dans le lac de données via la fenêtre illustrée dans la figure 5.15

Figure 5.15 : Data Lake Structure and Data.

L'utilisateur peut revenir vers le menu principal (Figure 5.14) en cliquant sur le bouton **« Explore More ».**

Pour que l'utilisateur explore la structure et les données d'une source de données, il doit d'abord commencer par explorer les types des sources de données qui alimentent le lac de données en cliquant sur le bouton **« sources types »** (Voir la figure 5.15).

- **Structure et Données d'une source relationnelle**

La figure 5.16 représente la structure de la source relationnelle **SE_MySQL** de notre système :

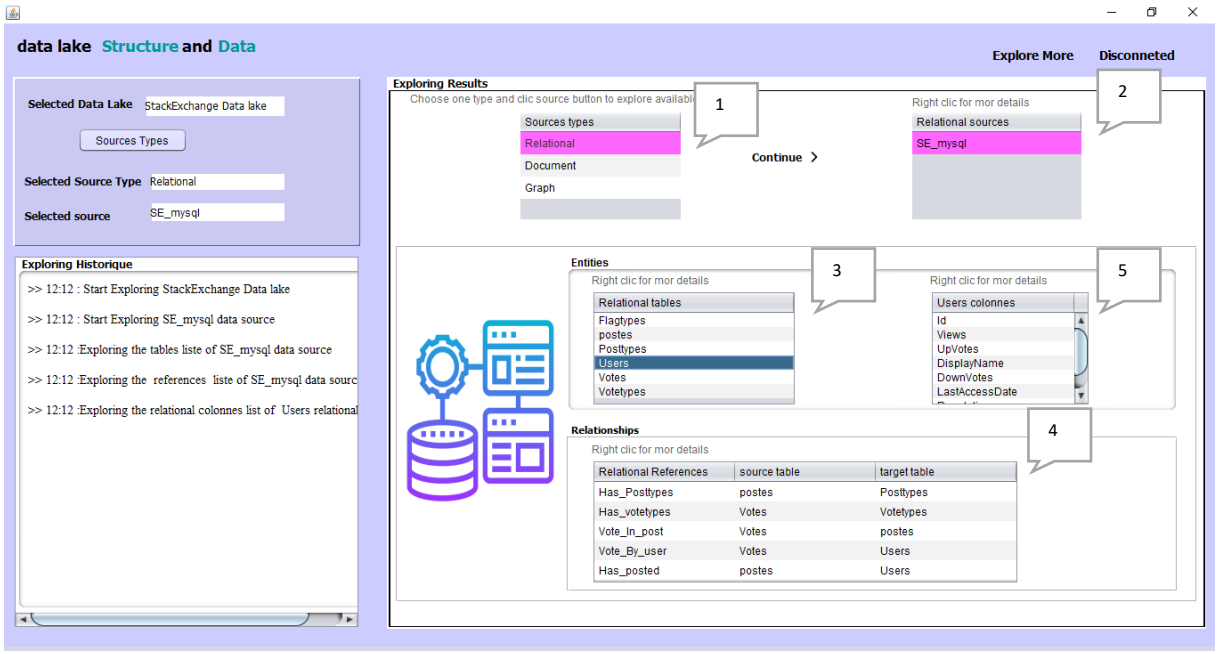
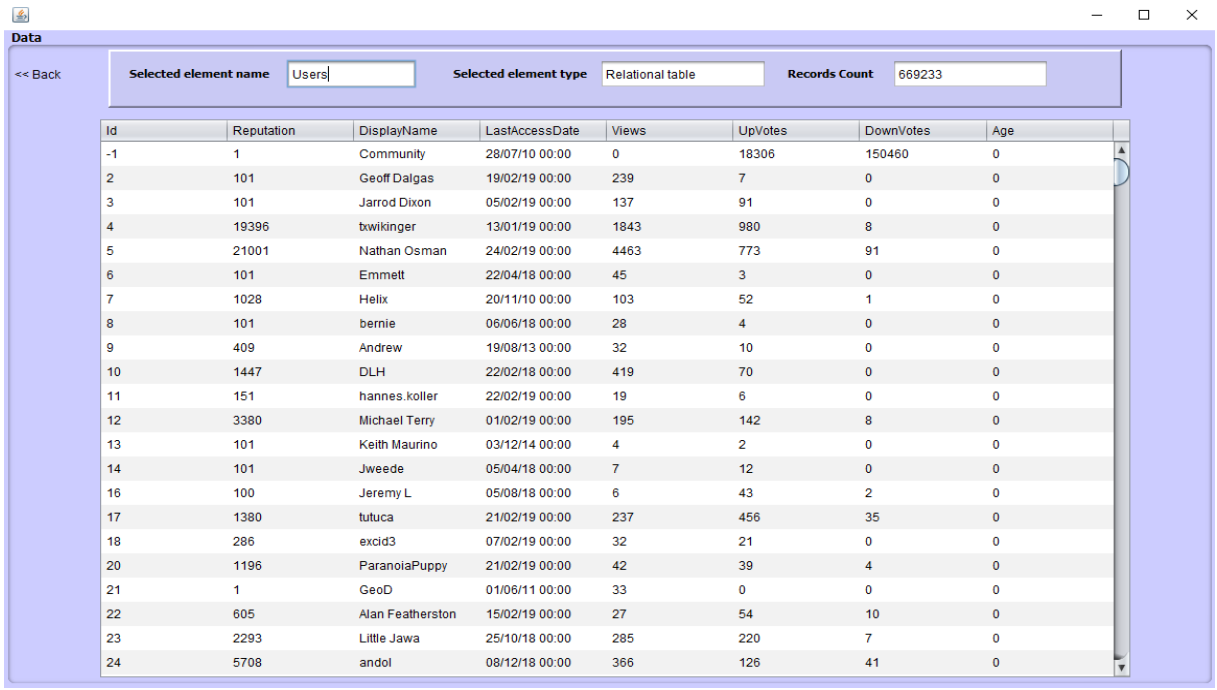


Figure 5.16 : Structure de la source SE_MySQL.

- **(1)** Liste des types de sources d lac de données StackExchange.
- **(2)** Liste des sources existant de type « Relational».
- **(3)** Liste des tables de la source relationnelle SE_MySQL. Dans cette étape l'utilisateur peut explorer les données d'une table relationnelle choisie, la figure 5.17 représente les données de la table « Users ».

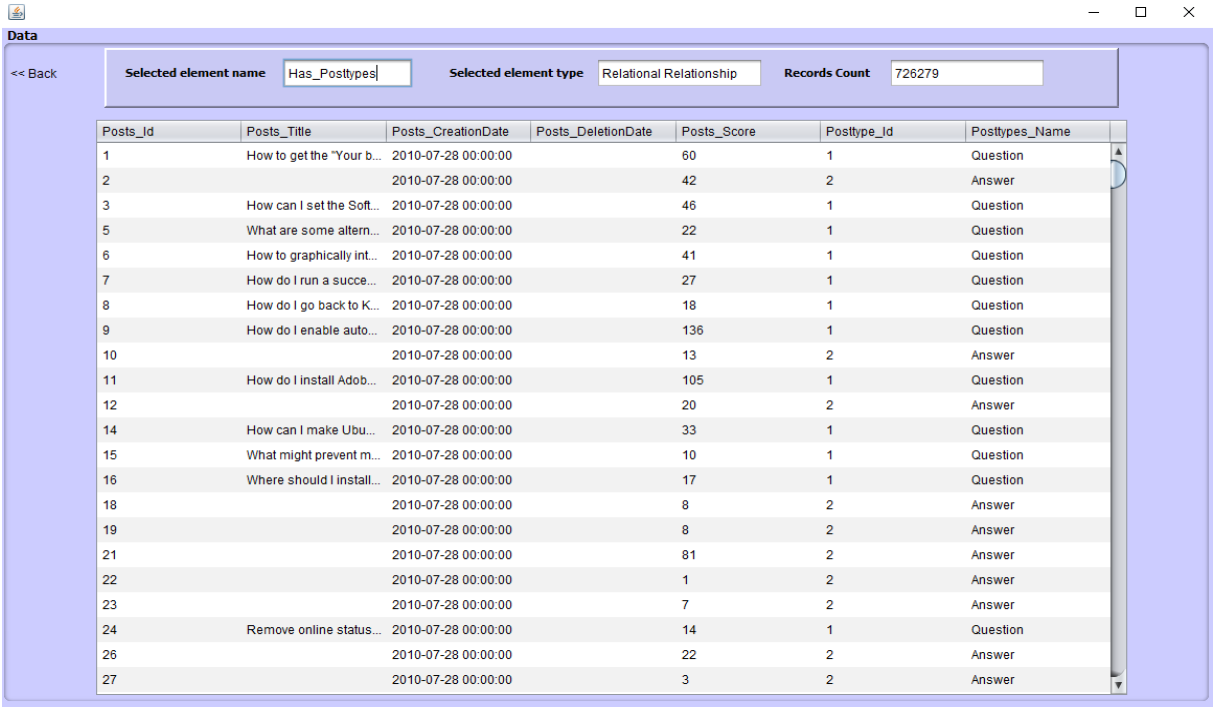


Id	Reputation	DisplayName	LastAccessDate	Views	UpVotes	DownVotes	Age
-1	1	Community	28/07/10 00:00	0	18306	150460	0
2	101	Geoff Dalgas	19/02/19 00:00	239	7	0	0
3	101	Jarrod Dixon	05/02/19 00:00	137	91	0	0
4	19396	txwikinger	13/01/19 00:00	1843	980	8	0
5	21001	Nathan Osman	24/02/19 00:00	4463	773	91	0
6	101	Emmett	22/04/18 00:00	45	3	0	0
7	1028	Helix	20/11/10 00:00	103	52	1	0
8	101	bernie	06/06/18 00:00	28	4	0	0
9	409	Andrew	19/08/13 00:00	32	10	0	0
10	1447	DLH	22/02/18 00:00	419	70	0	0
11	151	hannes.koller	22/02/19 00:00	19	6	0	0
12	3380	Michael Terry	01/02/19 00:00	195	142	8	0
13	101	Keith Maurino	03/12/14 00:00	4	2	0	0
14	101	Jweede	05/04/18 00:00	7	12	0	0
16	100	Jeremy L	05/08/18 00:00	6	43	2	0
17	1380	tutuca	21/02/19 00:00	237	456	35	0
18	286	excid3	07/02/19 00:00	32	21	0	0
20	1196	ParanoiaPuppy	21/02/19 00:00	42	39	4	0
21	1	GeoD	01/06/11 00:00	33	0	0	0
22	605	Alan Featherston	15/02/19 00:00	27	54	10	0
23	2293	Little Jawa	25/10/18 00:00	285	220	7	0
24	5708	andol	08/12/18 00:00	366	126	41	0

Figure 5.17 : Données de la table relationnelle « Users ».

- (4) Liste des références (associations entre les tables) relationnelles de la source relationnelle SE_MySQL. Dans cette étape l'utilisateur peut explorer les données

d'une référence relationnelle choisie. La figure 5.18 représente les données de la référence relationnelle «Has_Posttypes ».



Posts_Id	Posts_Title	Posts_CreationDate	Posts_DeletionDate	Posts_Score	Posttype_Id	Posttypes_Name
1	How to get the "Your b...	2010-07-28 00:00:00		60	1	Question
2		2010-07-28 00:00:00		42	2	Answer
3	How can I set the Soft...	2010-07-28 00:00:00		46	1	Question
5	What are some altern...	2010-07-28 00:00:00		22	1	Question
6	How to graphically int...	2010-07-28 00:00:00		41	1	Question
7	How do I run a succe...	2010-07-28 00:00:00		27	1	Question
8	How do I go back to K...	2010-07-28 00:00:00		18	1	Question
9	How do I enable auto...	2010-07-28 00:00:00		136	1	Question
10		2010-07-28 00:00:00		13	2	Answer
11	How do I install Adob...	2010-07-28 00:00:00		105	1	Question
12		2010-07-28 00:00:00		20	2	Answer
14	How can I make Ubu...	2010-07-28 00:00:00		33	1	Question
15	What might prevent m...	2010-07-28 00:00:00		10	1	Question
16	Where should I install...	2010-07-28 00:00:00		17	1	Question
18		2010-07-28 00:00:00		8	2	Answer
19		2010-07-28 00:00:00		8	2	Answer
21		2010-07-28 00:00:00		81	2	Answer
22		2010-07-28 00:00:00		1	2	Answer
23		2010-07-28 00:00:00		7	2	Answer
24	Remove online status...	2010-07-28 00:00:00		14	1	Question
26		2010-07-28 00:00:00		22	2	Answer
27		2010-07-28 00:00:00		3	2	Answer

Figure 5.18 : Données de la référence relationnelle « Has_Posttypes ».

- **(5)** Liste des colonnes de la table relationnelle « Users ». Dans cette étape l'utilisateur peut explorer les données d'une colonne relationnelle choisie.

- **Structure et Données d'une source Document**

La figure 5.19 représente la structure de la source document **SE_Mongodb** de notre système :

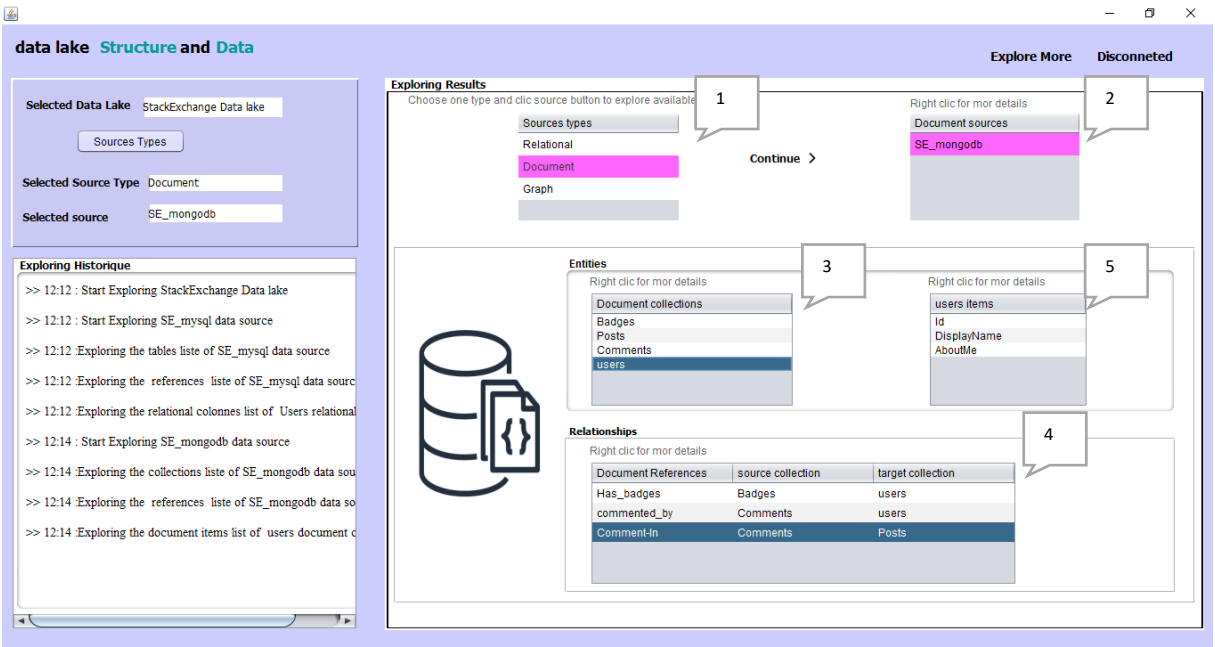


Figure 5.19 : Structure de la source SE_Mongodb.

- **(2)** Liste des sources existes de type « Document ».
- **(3)** Liste des collections de la source document SE_Mongodb. Dans cette étape l'utilisateur peut explorer les données d'une collection de documents choisie. La figure 5.20 représente les données de la collection «Users ».

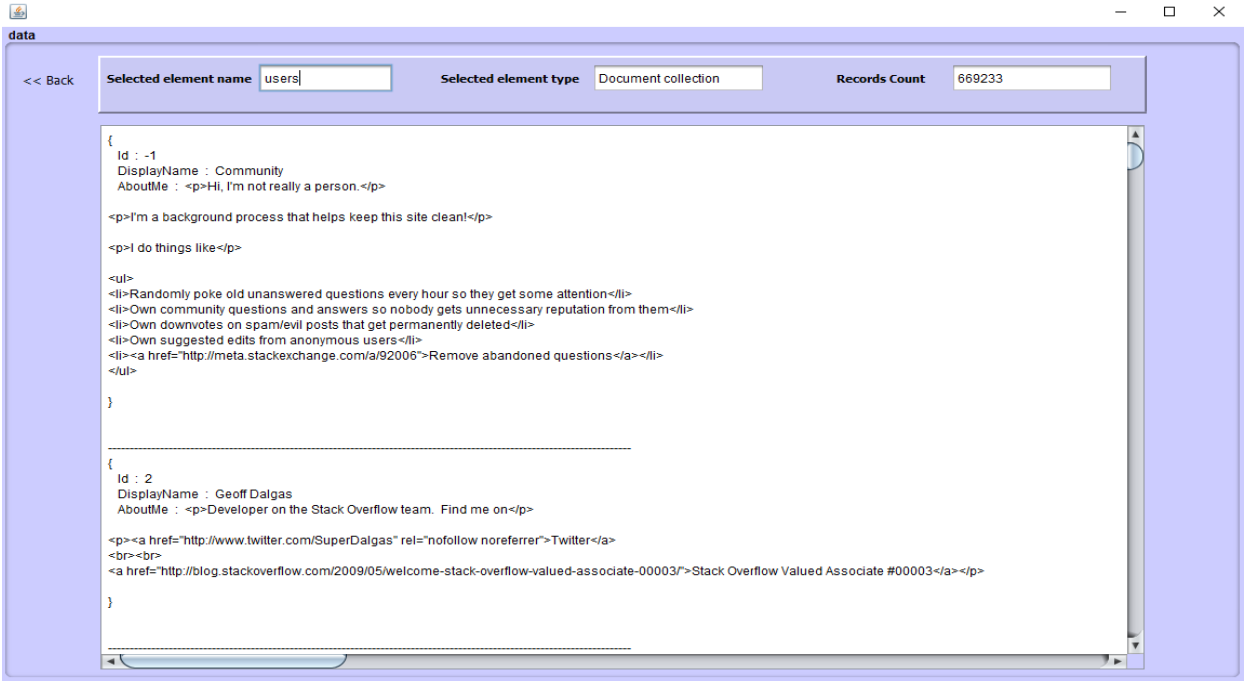


Figure 5.20 : Données de la collection « Users ».

- **(4)** Liste des références documents de la source document SE_Mongodb. Dans cette étape l'utilisateur peut explorer les données d'une référence choisie entre documents. La figure 5.21 représente les données de la référence « Commented_by ».

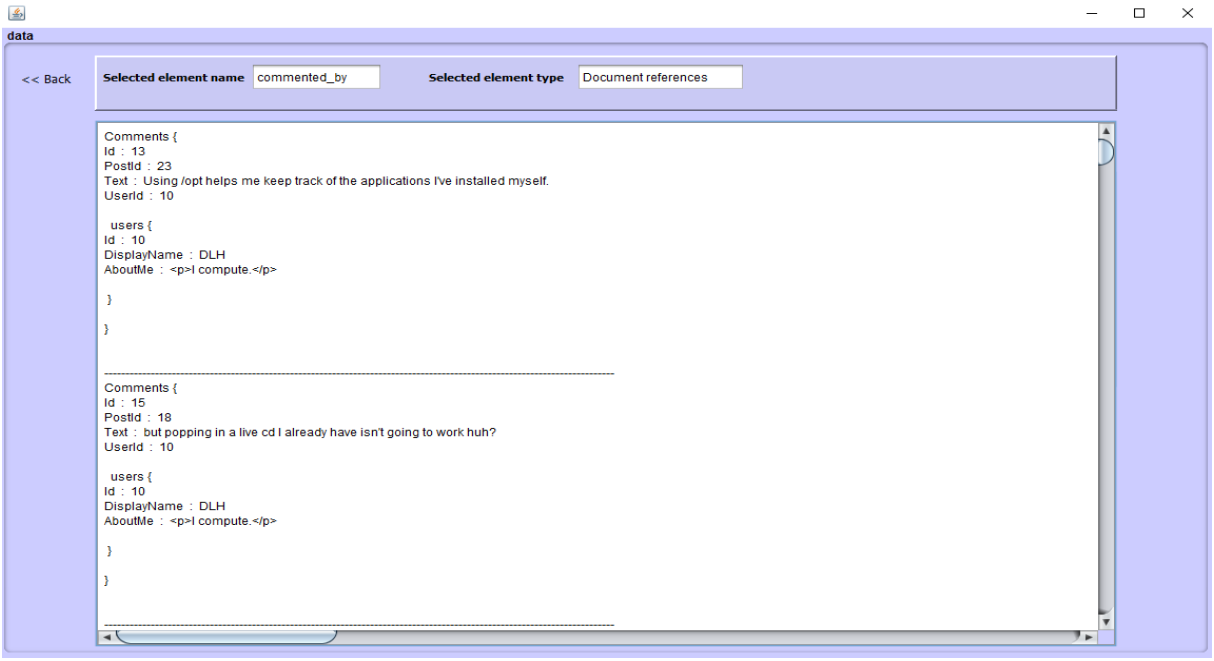


Figure 5.21 : Données de la référence document « commented_by ».

- **(5)** Liste des items de la collection « Users ». Dans cette étape l'utilisateur peut explorer les données d'un item de document choisi.

- **Structure et Données d'une source Graphe**

La figure 5.22 représente la structure de la source graphe **SE_Neo4j** de notre système :

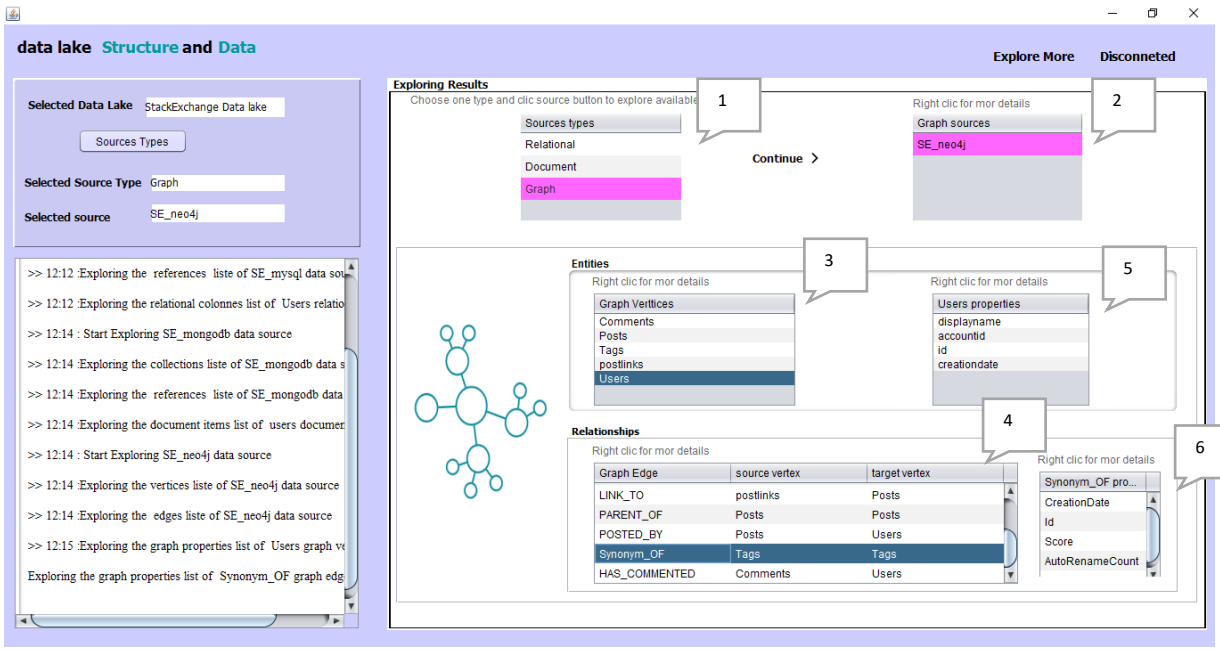


Figure 5.22 : Structure de la source SE_Neo4j.

- (2) Liste des sources existant de type « Graph ». Dans cette étape l'utilisateur peut récupérer la requête Cypher qui permet d'afficher les données d'une source graphe (voir la figure 5.23).

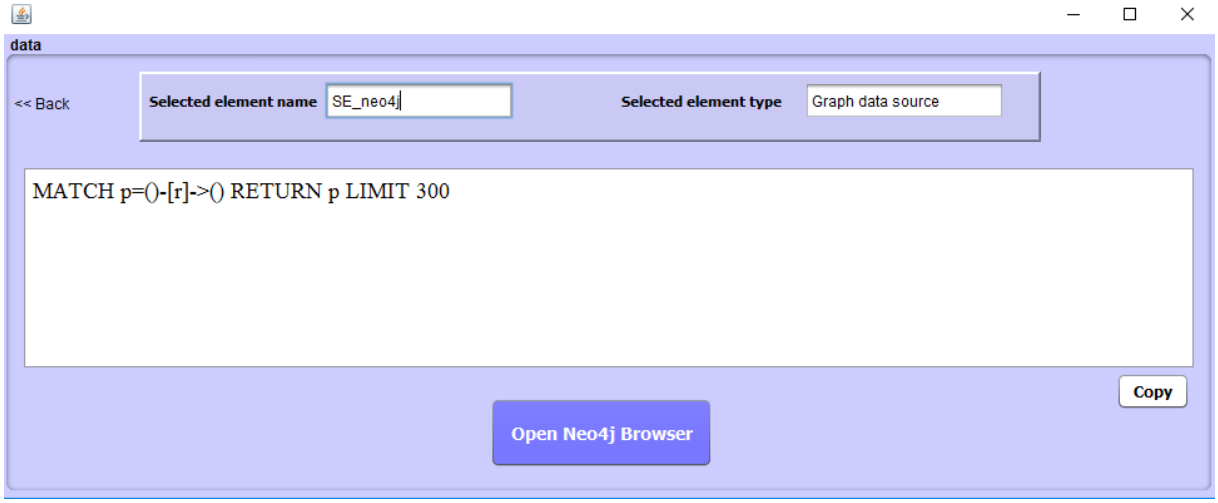


Figure 5.23 : Requête Cypher pour récupérer les données de la source graphe SE_Neo4j.

Pour que l'utilisateur puisse voir les données graphes à partir la requête récupérée, il doit la copier puis cliquer sur le bouton « Open Neo4j Browser », qui permet d'ouvrir le Neo4j Browser dans une page web. La figure 5.24 montre le résultat d'exécution de la requête récupérer dans la figure 5.23.

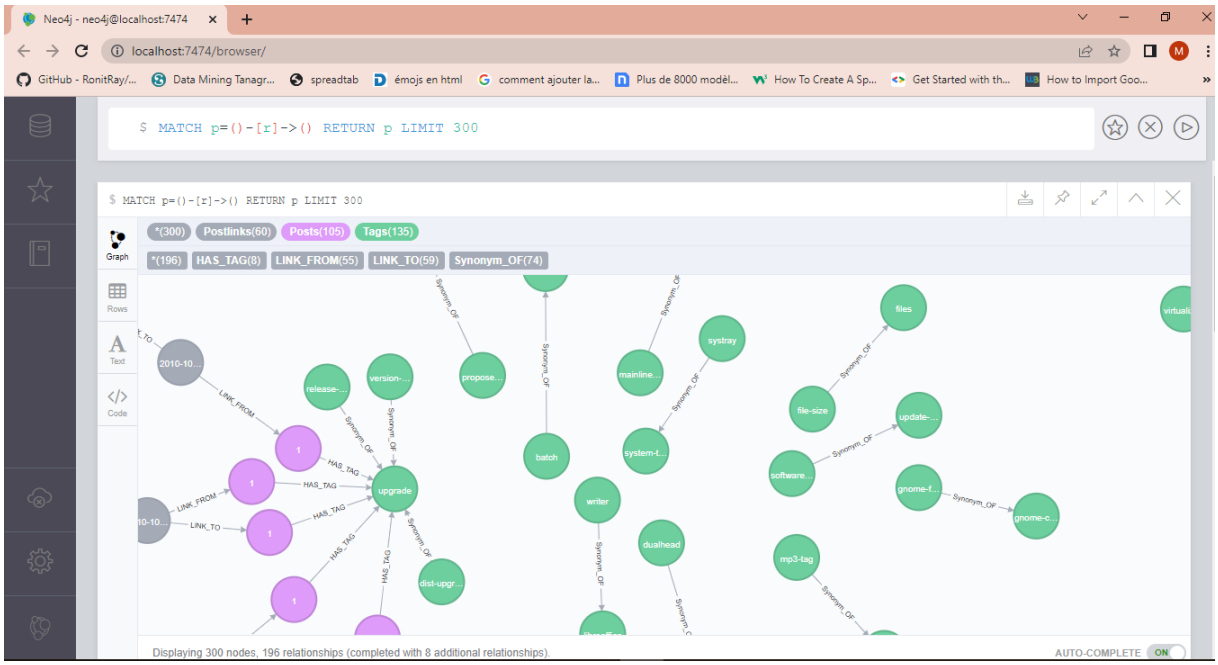


Figure 5.24 : Données de la source graphe SE_Neo4j dans Neo4j Browser.

- **(3)** Liste des vertices (nœuds) de la source graphe SE_Neo4j. Dans cette étape l'utilisateur peut récupérer la requête Cypher qui permet d'afficher les données d'un vertex. La figure 5 .25 représente la requête Cypher qui permet d'afficher les données de vertex Users.

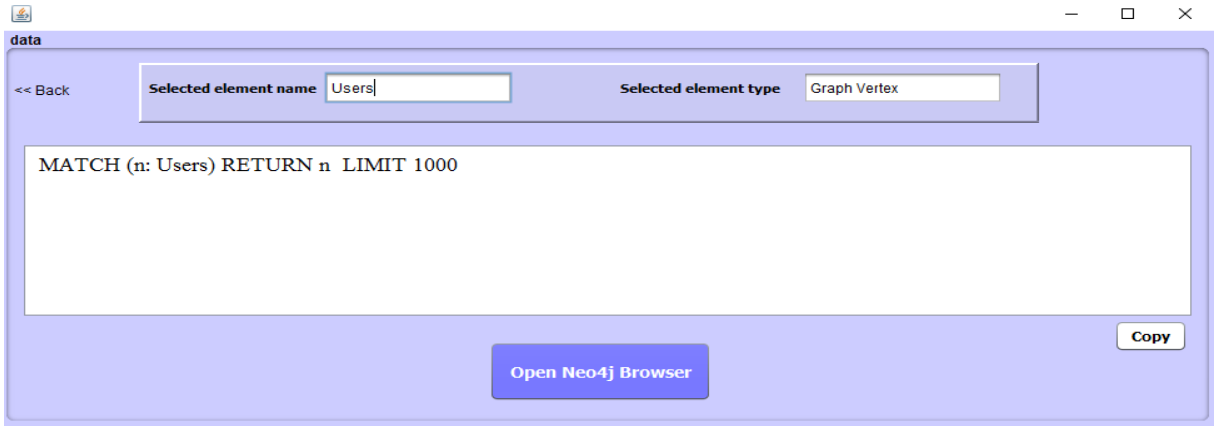


Figure 5.25 : Requête Cypher pour récupérer les données de vertex « Users».

- **(4)** Liste des edges (arcs) de graphes de la source graphe SE_Neo4j. Dans cette étape l'utilisateur peut récupérer la requête Cypher qui permet d'afficher les données d'un edge. La figure suivante représente la requête Cypher qui permet d'afficher les données de l'arc « POSTED_BY » :

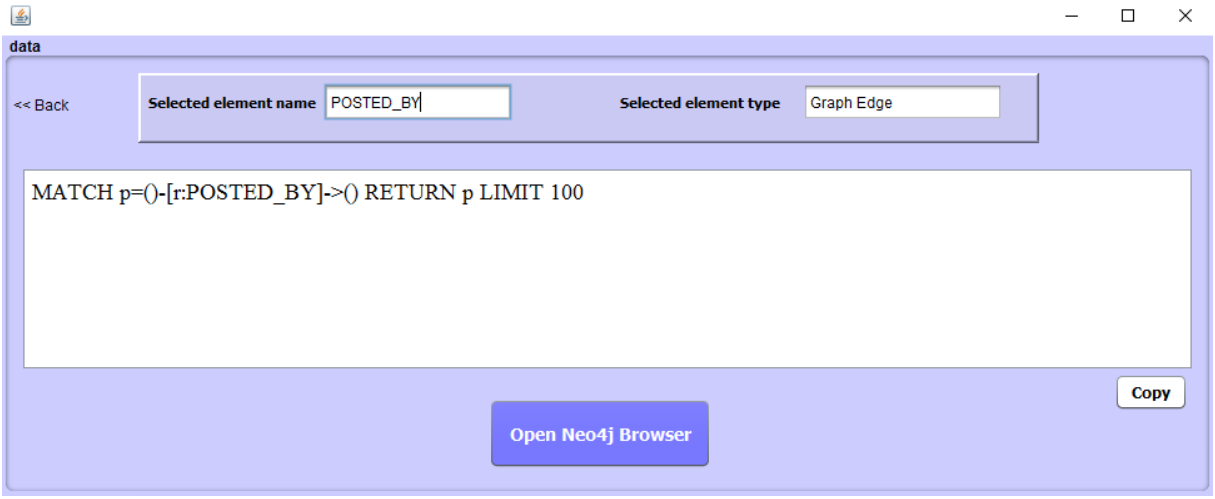


Figure 5.26 : Requête Cypher pour récupérer les données d'Edge « POSTED_BY ».

- **(5)** Liste des propriétés du vertex « Users ». Dans cette étape l'utilisateur peut récupérer la requête Cypher qui permet d'afficher les données d'une propriété d'un vertex.
- **(6**) Liste des propriétés du edge « Posted_By ». Dans cette étape l'utilisateur peut récupérer la requête Cypher qui permet d'afficher les données d'une propriété d'une Edge

– **Data Lake Metadata**

L'utilisateur peut explorer les Métadonnées d'un élément de lac de données via la fenêtre illustrée dans la figure 5.27

Figure 5.27 : Data Lake Meta data.

La figure suivante représente un exemple d'affichage des Métadonnées d'un élément de lac de données :

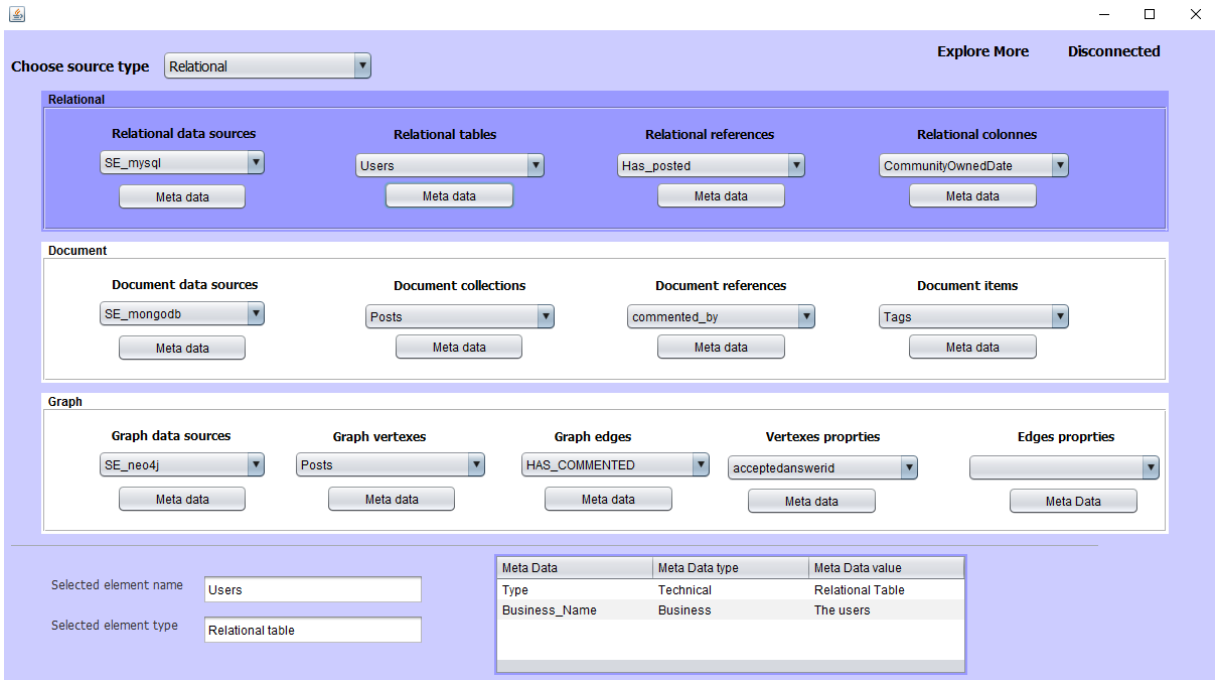


Figure 5.28 : Métadonnées de la table relationnelle « Users ».

– **Data Lake 360° View**

Avec cette fonctionnalité l'utilisateur est capable d'avoir une vue 360° à partir du lac de données exploré. Cette vue comprend les entités et les liens sémantiques ainsi que leurs données. La page principale de cette fonctionnalité est illustrée dans la figure suivante :

Figure 5.29 : Data Lake 360° View.

La figure 5.30 montre un exemple d'une vue 360° sur l'entité sémantique « The users » :

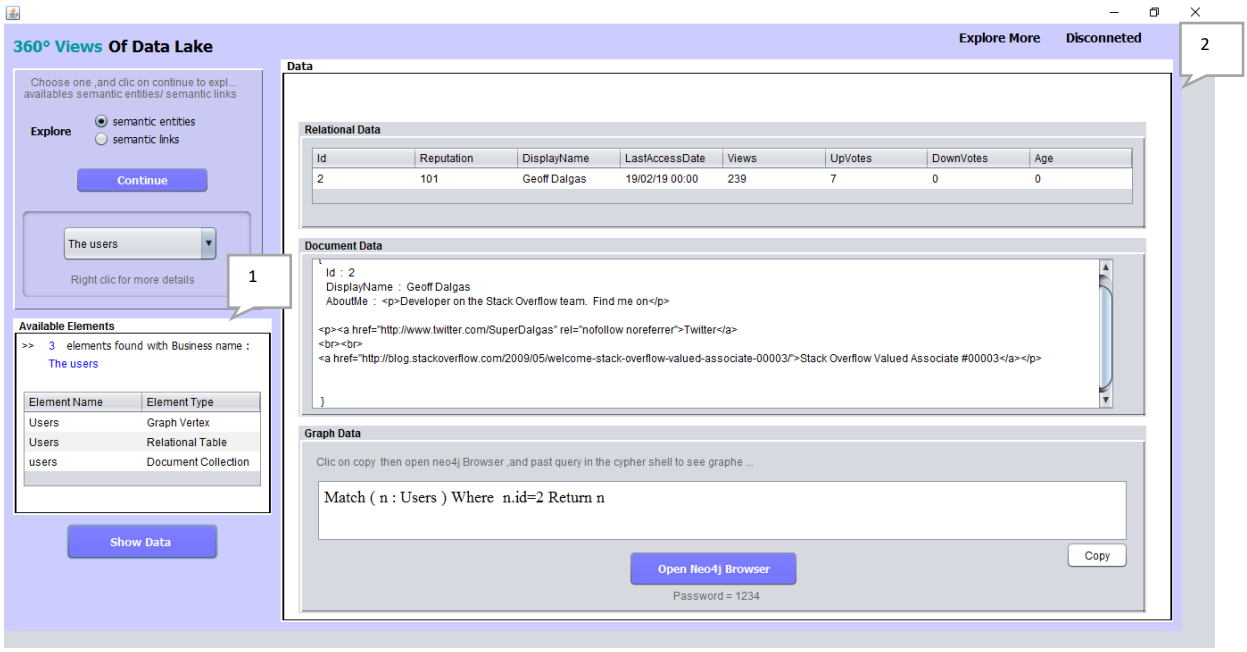


Figure 5.30 : Vue 360° sur l'entité « The users ».

- **(1)** L'ensemble des éléments du lac de données qui correspondent à l'entité sémantique « The users ».
- **(2)** Un exemple d'une vue 360° sur les données de l'entité « The users », ayant l'identifiant "2" qui doit être entré par l'utilisateur via la boîte dialogue illustré dans la figure 5.31.

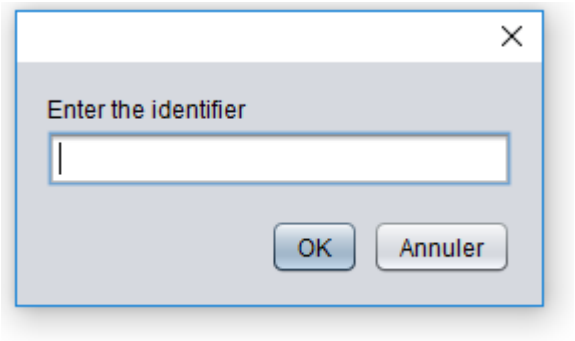


Figure 5.31 : Boite dialogue pour entrer l'identificateur.

La figure 5.32 montre une vue 360° sur le lien sémantique « The post posted by user ».

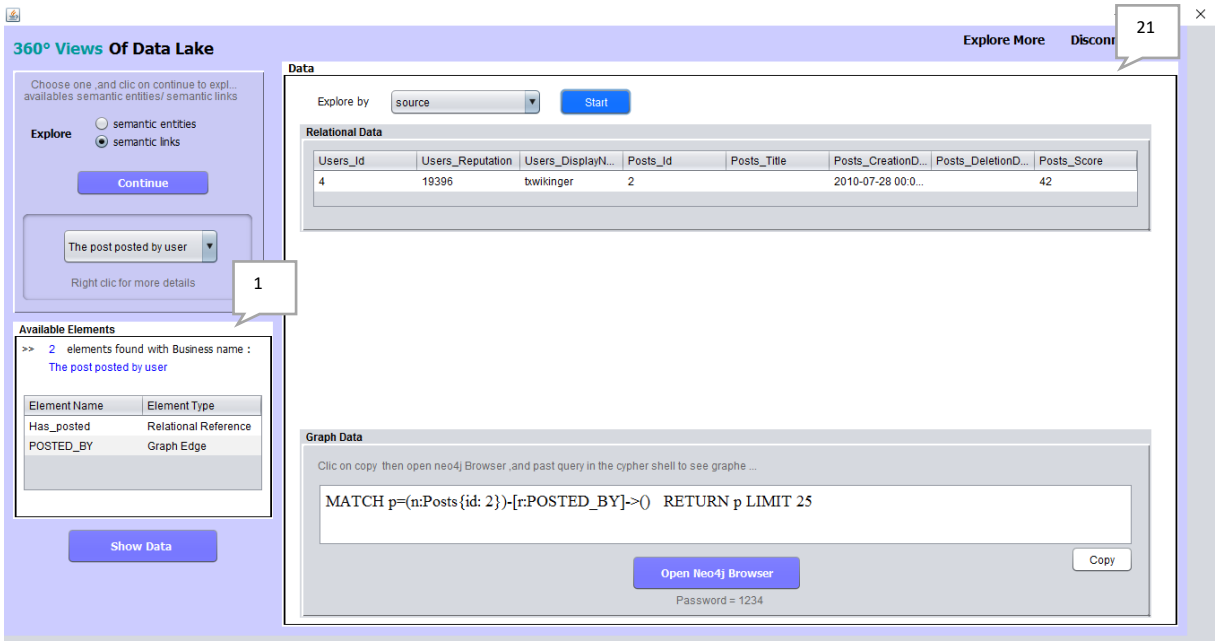


Figure 5.32 : Vue 360° sur le lien « The post posted by user ».

- **(1)** L'ensemble des éléments du lac de données qui correspondent au lien sémantique « The post posted by user ».
- **(2)** Un exemple d'une vue 360° sur les données du lien « The post posted by user », avec l'identificateur de l'entité source égale à 2, notant que l'utilisateur peut également explorer les données d'un lien sémantique par l'entité Target.

– **Search in Data Lake**

L'utilisateur peut effectuer une recherche simple sur le lac de données par un mot clé, ou une recherche avancée par une propriété de Métadonnées. La figure 5.33 représente le résultat d'une recherche simple avec le mot clé « user », tandis que la figure 5.34 montre le résultat d'une recherche avancée avec la propriété « data type » et la valeur « String », et la figure 5.35 représente le résultat de la recherche lorsqu'aucun élément de lac de données correspond au mot clé entré par l'utilisateur.

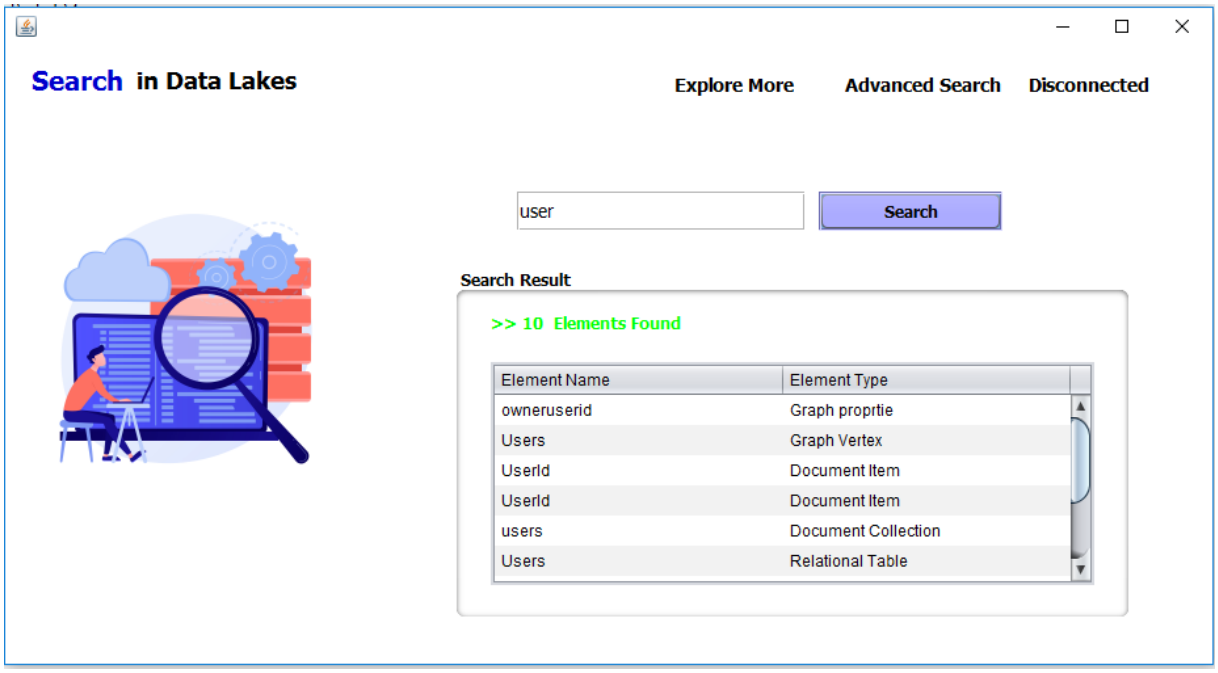


Figure 5.33 : Recherche simple avec le mot clé « User ».

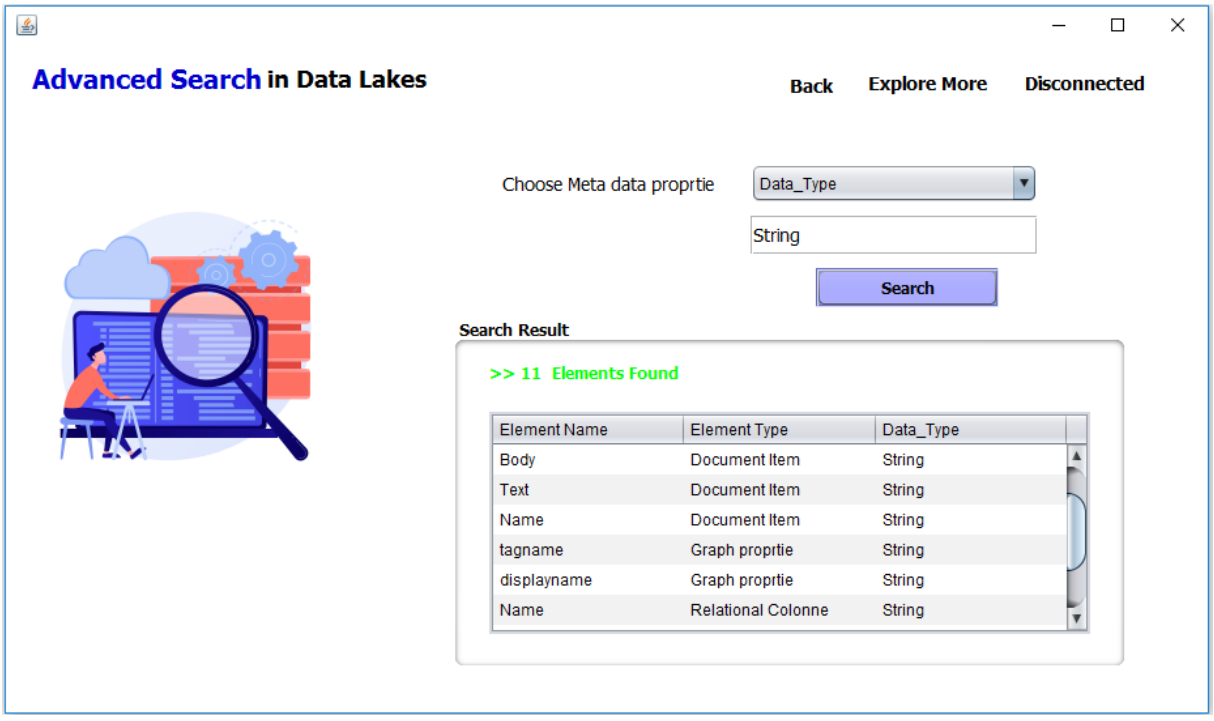


Figure 5.34 : Recherche avancée avec la de propriété de Métadonnées « data type » et la valeur « String ».

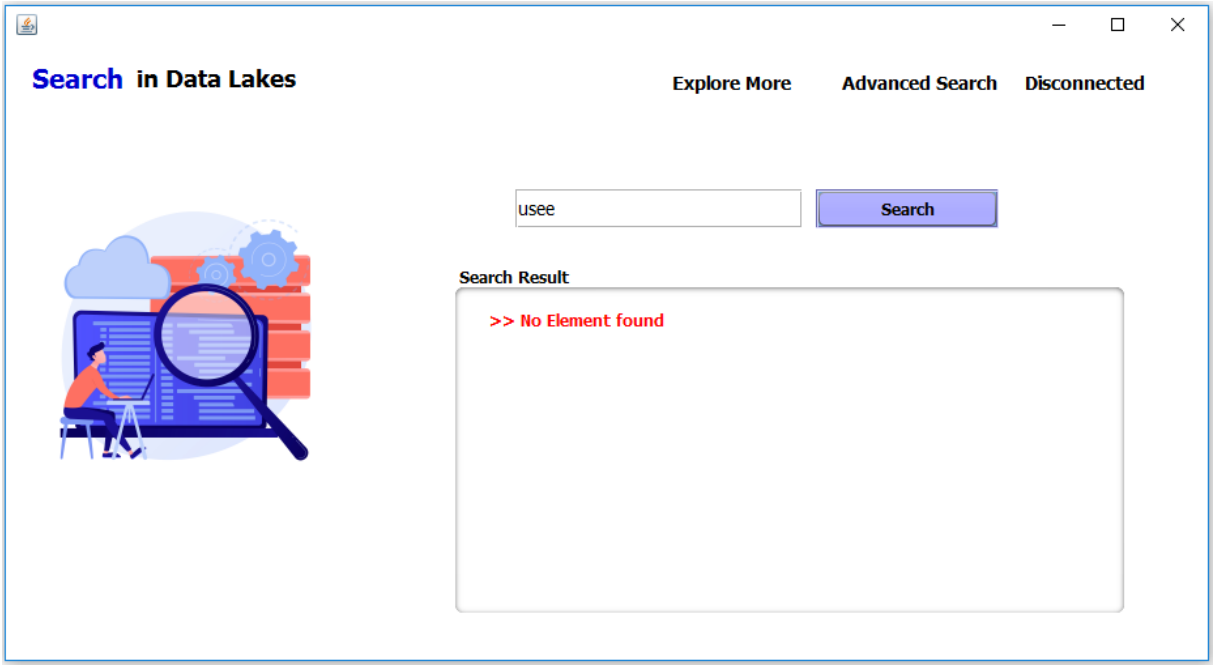


Figure 5.35 : Aucun résultat trouvé avec le mot clé « usee ».

– **About Us**

Les utilisateurs peuvent en savoir plus sur l'application. La figure 5.36 représente la page « About Us » de notre application :

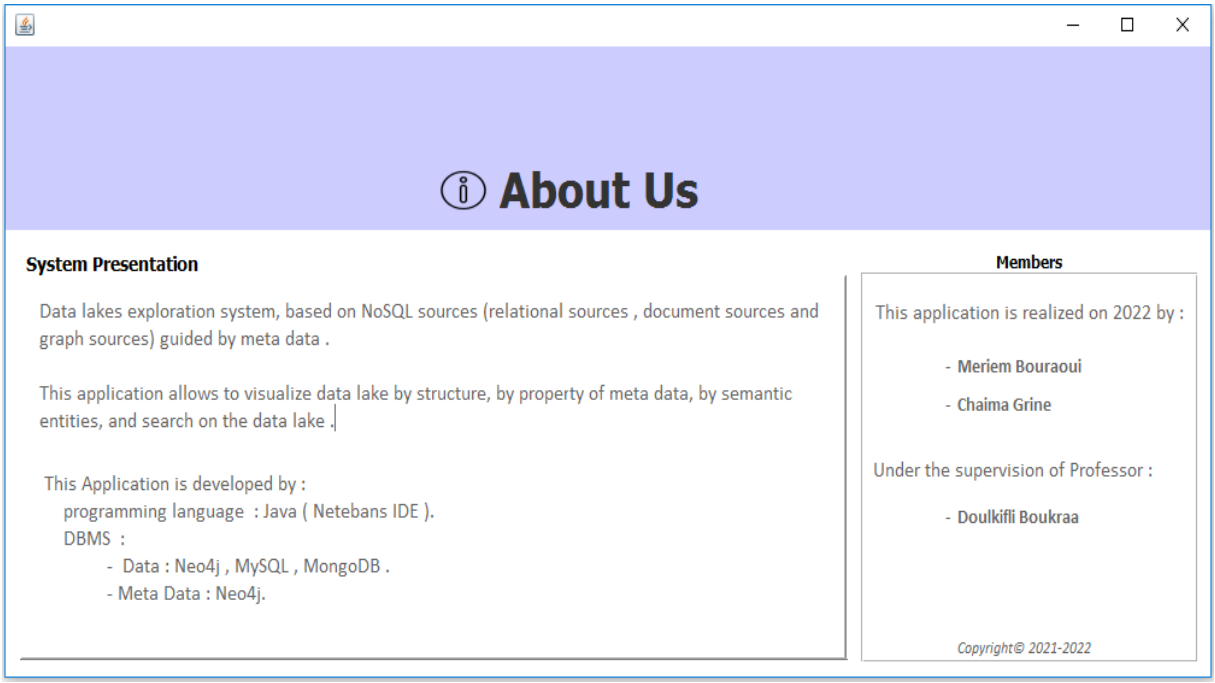


Figure 5.36 : La page « About Us » de notre application.

5.6 Conclusion

Dans ce chapitre, nous avons présenté l'implémentation d'un système d'exploration de lac de données. Nous avons commencé par une brève présentation de l'environnement de travail et les différentes technologies utilisées pour le développement. Ensuite, nous avons décrit le jeu de données utilisé et la manière de la reconstruction des bases de données de notre application à partir de la base de données de réseau StackExchange. Enfin, nous avons présenté quelques captures d'écrans qui illustrent les différentes facettes de notre application.

Conclusion Générale

Notre travail a été réalisé dans le cadre d'un projet de fin d'étude, et qui avait comme objectif de développer un système d'exploration de lacs de données composés des sources de données NoSQL. Ce système vise à résoudre le problème de l'incapacité à découvrir et explorer tout le contenu d'un lac de données composé des données de différents types, collectées à partir des diverses sources de données.

Afin de réaliser ce travail, nous avons présenté d'abord les notions de base : le Big Data, les bases de données NoSQL, les lacs de données, le mécanisme d'exploration des lacs de données ainsi que les métadonnées et leur rôle particulier dans ce mécanisme. Par la suite, nous avons abordé la capture de besoins fonctionnels du système ainsi que son analyse et sa conception en suivant le processus 2TUP d'une manière allégée qui sert les besoins de notre système.

Cette expérience nous a permis d'acquérir des compétences et d'enrichir nos connaissances et de découvrir de nouvelles technologies, notamment les deux concepts Big Data et Data Lake qui évoluent avec une vitesse très exceptionnelle.

Nous espérons que cette application puisse répondre aux exigences des usagers, et simplifier leurs tâches d'exploration. Comme perspective future, nous souhaitons améliorer notre application, par exemple introduire le mécanisme de la recherche sémantique, ou créer des nouvelles extensions de cette application avec d'autres types de sources de données, mais ce travail reste prêt pour toute amélioration ou extensions envisageables par d'autres personnes.

Bibliographie

[1] A. Amrane, “ Big Data Concepts et Cas d’utilisation ”, Centre de recherche sur l’information scientifique et Technique CERIST.
[2] P. Delort, “LE BIG DATA”, 1er edition, p. 128, 2015.
[3] P. Laflamme, “ Big Data et ses technologies”, ETS Montréal, 2017.
[4] R. Ouhab, “ Développement d’un système de Gestion des métadonnées dans les data lakes à base de source NoSQL”, Université Mohamed Sadik Benyahia – Jijel, 2020-2021.
[5] A. Harrou, Y. Ait Ali Yahia, “ Integration de données dans un contexe Big Data”, Université A. Mira-Béjaia, Algérie, 2019-2020.
[6] A. Amrane, “ Big Data Concepts et cas d’utilisation”, doi :10.13140/2.1.3076.2081, 2015.
[7] N. Tahreem, “ Comprendre les données structurées, semi-structurées et non-structurées”, 2020, (consulté : 28/02/2022).
[8] E. Laxmi Lydia, “ Big Data Analysis using Hadoop components like Flume, MapReduce, Pig and Hive ”, International Journal of Computer Science Engineering and Technology IJCSET, vol. 5, no. 11, pp. 390-394, 2015.
[9] Y. Saadna, “Cours Big Data et deep learning”, Université Batna 2, 2019-2020.
[10] TechTarget : https://whatis.techtarget.com/fr/definition/In-Memory-Base-de-donnees-en-memoire (consulté : 28/02/2022).
[11] L. Soulier, “ Introduction BIG DATA”, université de pierre et marie curie, France, 2016-2017.
[12] J.P. Lajonchère, “ Le rôle des Méga données dans l’Evolution de la pratique Médicale”, Académie nationale de médecine, Paris, pp. 225-240, 2018.
[13]A.Davoust , “ La gestion des « Big data » en marketing : défis, opportunités et voies de recherche” ,Université Paris-Dauphine, 2014.
[14] DataScientest : https://datascientest.com/nosq (consulté : 01/03/2022).
[15] S. Gilbert, NA. Lynch, “ Prespectives on the CAP Theorem”, IEEE COMPUTER SOCIETY, vol.45, pp. 30-36, 2012.
[16] I. Benmia, “ Migration d’une base de données relationnelle Vers une base de données distribuée NoSQL”, Université Abou Bakr Belkaid– Tlemcen, 2014-2015.
[17] OpenClassroom : https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4462471-maitrisez-le-theoreme-de-cap (consulté : 02/03/2022).
[18] S.Fremigier, “ BIG DATA OPEN SOURCE: UNE CONVERGENCE INVÉITABLE?”, 2012.
[19] H. Hashem, “ Le traitement BigData”, Edition Publishroom, p. 324,2021.
[20] Z. BenAllal, H.T ahraoui, “ Etude comparative des bases de données NoSQL : MongoDb, CouchBase, Cassandra, HBase, Redis, OrientDB ”, Université Abou Bakr Belkaid– Tlemcen,2015-2016.

[21] R.Angles, C. Gutierriez, " Survey Of Graph Data Base Models", Université de chili, vol. 40, no. 1, 2008.
[22] A. Donkers, D. Yang, N. Baken, " Linked Data for Smart Homes : Comparing RDF and Labeled Property Graphes", 2020.
[23] R. Angles, " the property Graph Database Model", Université de talca,2018.
[24] E. Jebri, " PROPOSITION D'UNE NOUVELLE MÉTHODE DE GÉNÉRATION DE REQUÊTES SPARQL A PARTIR DE GRAPHES RDF ", Université de Tunis El MANAR, 2017-2018.
[25] A.G. Piazza, " NOSQL", Haute École de Gestion de Genève (HEG-GE),2013.
[26] N. Vergnes, " Bases de données graphes : Comparaison de NEO4J et OrientDB", Maison de la recherche et de la valorisation IPST-CNAM, 2015.
[27] J. Webber, L. Robinson, " The Top 5 Use Cases of Graph Databases….",neo4j.com (consulté: 02/03/2022).
[28] Oxford learner'sDictionaries : https://www.oxfordlearnersdictionaries.com/ (consulté :08/03/2022).
[29] R. Tighilt Ferhat, " Extraction des modèles d'une base de données NoSQL orientéedocuments basée sur une approche dirigée par les modèles", 'Université Toulouse 1 Capitole, 2021.
[30] R. Arora, R.R. Aggarwal, "Modeling and Query Data in MongoDB", International Journal of Scientific & Engineering Researche, vol. 4, no. 7,2013.
[31] A. Wiliam, "NoSQL database type explaind: Document-based databases", https://www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Document-based-databases (consulté : 8/3/2022).
[32] Amazon: https://docs.aws.amazon.com/documentdb/latest/developerguide/document-database-use-cases.html (consulté : 8/3/2022).
[33] E. Scholly, P. Saxadogo, C. Favre, E. Ferey et all, " Système de méta données dans les lacs de données: modélisation et fonctionnalités", Université de Montepeller, 2019.
[34] C. Madera, A. Laurent. "The Next Information Architecture Evolution: The Data Lake Wave", MEDES: Management of Digital Eco Systems, 2016.
[35] F. Ravat,Y. Zhao, "Data Lakes: Trends and Perspectives. ", International Conference on Database and Expert Systems Applications (DEXA 2019), pp.304-313,hal-02397457, 2019,
[36] B. Stein, A. Morrison, "The entreprise data lake : Better integration and deeper analytics",2014.
[37] R. Kafando, R. Decoupes, L. Sautot, " Maguelonne Teisseire. Modélisation de la dynamique des territoires : métadonnées et lacs de données dédiés à l'information spatiale", INFORSID2020, pp. 1-16,hal-02947913, 2020.
[38] C. Madera, " L'évolution des systèmes et architectures d'information sous l'influence des données massives : les lacs de données", Base de données [cs.DB], Université Montpellier, 2018.
[39] P. Sawadogo, J .Darmont. " On data lake architectures and metadata management". Journal of Intelligent Information Systems, vol 56 , no 1, pp.97-120. Doi 10.1007/s10844-020-00608-7. hal-03114365
[40] M. Haroon, "Big Data Exploration, Visualization and analytics", Atlantic Business centre, 2020.

[41] H. Kadi, “Exploration des données de la médecine personnalisée par des techniques de data mining”, Université Mustapha Stambouli, 2021-2022.
[42] A.DJEFFAL ,“ Cours Fouille de données avancée”, Université Mohamed Khider – Biskra, 2015-2016.
[43] R. Hai, Ch. Quix, M. Jarka, “Data Lake Concept and Systems: a Survey”,2021.
[44]TechTarget : :https://www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Graph (consulté : 07 /04 /2022).
[45] P. Roques , F. Vallée, “UML 2 en action De l’analyse des besoins à la conception”,4e édition p.382,2007.
[46] Wikipedia : https://fr.wikipedia.org/wiki/Java (consulté: 02/06/2022).
[47]Ibracilinks: https://ibracilinks.com/blog/quand-et-pourquoi-java-est-utilise-pour-le-developpement-dapplications#:~:text=Java%20est%20un%20langage%20de,%2C%20le%20support%20communautaire%2C%20etc (consulté: 02/06/2022).
[48]Developpement-informatique:https://developpement-informatique.com/article/233/differences-entre-jdk-jre-et-jvm (consulté: 02/06/2022).
[49]Oracle:https://www.oracle.com/dz/tools/technologies/netbeans-ide.html(consulté:22/06/2022)
[50]Sql.Sh: https://sql.sh/sgbd/mysql#:~:text=Caract%C3%A9ristiques,NET%2C%20Python%20%E2%80%A6(consulté: 02/06/2022).
[51] R. Henricsson, “Document Oriented NoSQL Databases A comparison of performance in MongoDB and CouchDB using a Python interface”, Blekinge Institute of Technology, 2011
[52] A. Holemans, “Implémentation d’une base de données NoSQL de données géospatiales de l'AIDE", Université de LIÈGE, 2016-1017.
[53]https://www.infoq.com/fr/profile/Peter-Neubauer/P. Neubauer, “Les Bases Orientées Graphes, NoSQL et Neo4j ",pp.1-18, 2010