AVERTISSEMENT

LIENS

Code la Propriété Intellectuelle – Articles L. 122-4 et L. 335-1 à L. 335-10
Loi n° 92-597 du 1er juillet 1992, publiée au *Journal Officiel* du 2 juillet 1992

http://www.cfcopies.com/V2/leg/leg-droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

L’université n’entend ni approuver ni désapprouver les opinions particulières du candidat.

Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du
DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE
Délivré par l'Université Toulouse 1 Capitole

Présentée et soutenue par

Rym JEMMALI

Le 13 décembre 2023

Processus d'ingestion de données hétérogènes et d'assistance au requêtage pour un lac de données médical

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Gilles ZURFLUH et Fatma ABDELHEDI

Jury

M. Omar BOUSSAID, Rapporteur
M. Slimane HAMMOUDI, Rapporteur
Mme Faten ATIGUI, Examinatrice
M. Elhadj BENKHELIFA, Examinateur
M. Gilles ZURFLUH, Directeur de thèse
Mme Fatma ABDELHEDI, Co-directrice de thèse
M. Lionel RIGAUD, Invité

Cette thèse a été réalisée dans le cadre de la convention CIFRE N° 2020/0526 entre la société TRIMANE et l'Université Toulouse Capitole. Les travaux ont été réalisés au sein du laboratoire IRIT de Toulouse.

La thèse a été soutenue à l'Université Toulouse Capitole le 13 décembre 2023 devant le jury suivant :

Rapporteurs :

M. Omar BOUSSAID, Professeur à l'Université Lyon 2

M. Slimane HAMMOUDI, Enseignant chercheur à l'ESEO d'Angers

Directeurs :

Mme Fatma ABDELHEDI, Directrice du laboratoire CBI^2 de TRIMANE à Puteaux

M. Gilles ZURFLUH, Professeur à l'Université Toulouse 1 Capitole

Examinateurs :

Mme Faten ATIGUI, Maitre de conférences au Conservatoire national des arts et métiers de Paris

M. Elhadj BENKHELIFA, Full Professor at Staffordshire University (Angleterre)

Invité :

M. Lionel RIGAUD, Directeur général à la société TRIMANE à Puteaux

Résumé

Les avancées technologiques récentes ont permis une explosion des données générées à une échelle sans précédent, ce que l'on appelle communément le « Big Data ». Les entreprises, les organisations et même les particuliers sont confrontés à des volumes massifs de données provenant de sources diverses telles que les réseaux sociaux, les capteurs « IoT », les transactions en ligne, les appareils mobiles, etc. Les techniques standards de traitement, de stockage et d'analyse des données ont été reconsidérées et étendues, voire redéfinies pour prendre en compte des contraintes inhérentes à ce domaine d'étude.

Dans ce contexte, les lacs de données ont émergé comme une solution prometteuse pour le stockage et l'exploitation de mégadonnées (Big Data), en complément aux entrepôts de données. Un lac de données se définit par deux propriétés principales : la variété des données qu'il est capable d'ingérer, et une approche où le schéma des données n'est défini qu'à leur interrogation (*schema-on-read*). Ces propriétés font qu'un lac de données est un système souple et adaptatif.

Nos travaux s'inscrivent dans le cadre d'une convention CIFRE dont le projet est de développer une application concernant l'exploitation d'un lac de données médical pour le compte de mutuelles santé. Dans ce mémoire, nous proposons une solution pour permettre à des acteurs-métier (non-informaticiens) de manipuler un lac de données existant.

Cependant, l'hétérogénéité des systèmes de stockage associée à la diversité des sources du lac de données constitue un obstacle majeur à une exploitation efficace des données. Les outils de stockage actuels offrent peu de mécanismes pour prendre en compte l'hétérogénéité des sources stockées dans le lac de données tout en garantissant la cohérence des données.

Notre première contribution porte sur l'ingestion des données à partir d'un lac de données en vue de créer une BD appelée entrepôt. L'ingestion consiste, tout d'abord, à transférer des BD relationnelles et NoSQL extraites du lac de données dans une base de données NoSQL unique (l'entrepôt). Pour automatiser ce processus, nous avons utilisé l'architecture MDA (Model Driven Architecture) qui offre un environnement de transformation des schémas. A partir des schémas physiques décrivant un lac de données, nous proposons des règles de transformation qui permettent de créer un entrepôt de données stocké sous un système NoSQL orienté-documents.

Cet entrepôt, provenant de l'ingestion de données massives, présente une structure complexe. À ce titre, nous proposons un processus d'assistance aux acteurs-métier pour leur faciliter l'accès aux informations et l'observation de leurs activités. Ce processus repose soit sur un mécanisme de personnalisation soit sur un mécanisme de recommandation qui assistent les acteurs-métier dans leur recherche d'information.

Une expérimentation a été réalisée pour une application médicale destinée à une mutuelle de santé.

Mots-clés : informatique décisionnelle, lac de données, entrepôt de données, magasins de données, Big Data, base de données relationnelle, base de données NoSQL, MDA, mécanisme de recommandation, personnalisation.

Abstract

Recent technological advances have led to an explosion of data generated at an unprecedented scale, commonly referred to as "Big Data". Businesses, organizations, and even individuals are faced with massive volumes of data from a variety of sources such as social media, IoT sensors, online transactions, mobile devices, etc. Standard data processing, storage, and analysis techniques have been reconsidered and extended, or even redefined to take into account the inherent constraints of this field of study.

In this context, data lakes have emerged as a promising solution for storing and exploiting big data, in addition to data warehouses. A data lake is defined by two main properties: the variety of data it can ingest, and an approach where the data schema is defined only at query time (*schema-on-read*). These properties make a data lake a flexible and adaptive system.

However, the heterogeneity of storage systems combined with the diversity of data lake contents is a major obstacle to effective decision-making exploitation of data. In addition, traditional decision support systems cannot meet the growing demands of modern businesses for the integration and analysis of the massive amounts of data generated. Thus, it is essential to reorganize this data into a unified form. Current storage tools offer few mechanisms to take into account this heterogeneity of databases while ensuring data consistency and, consequently, their quality. Most organizations must therefore transform data stored in relational systems into NoSQL or "Not only SQL" systems based on flexible models.

In this paper, we propose solutions to allow decision-makers to manipulate complex data stored in heterogeneous databases. Our first contribution concerns the ingestion of data from a data lake in order to create a database called warehouse and intended for decision-making analysis. Ingestion consists, first, of transferring relational and NoSQL BDs extracted from the data lake into a single NoSQL database (the warehouse), then merging so-called "similar" classes, and finally converting links into references between objects. To automate this process, we used the MDA (Model Driven Architecture) architecture which offers a schema transformation environment. From the physical schemas describing a data lake, we propose transformation rules that allow us to create a data warehouse stored under a document-oriented NoSQL system.

This warehouse, resulting from the ingestion of massive data, has a complex structure. As such, we propose a user assistance process to facilitate their access to data. This process is based on either a personalization system or a recommendation system that assists decision-makers in their information retrieval. This process is based on either a personalization system or a recommendation system that assists decision-makers in their information retrieval.

An experiment was conducted for a medical application intended for a health insurance company.

Keywords: Business Intelligence, Data Lakes, Data Warehouses, Data Marts, Big Data, relational databases, NoSQL databases, MDA, recommendation systems, personalization.

Remerciements

En préambule de ce manuscrit, j'aimerais exprimer ma reconnaissance à tous ceux qui ont contribué, de près ou de loin, à la réalisation de cette thèse et à mon développement en tant que doctorante.

Avant tout, je voudrais exprimer ma profonde et sincère gratitude à mon directeur de thèse, monsieur Gilles ZURFLUH, Professeur à l'Université Toulouse 1 Capitole pour la qualité de son encadrement, sa totale disponibilité à mon égard, pour son aide, ses encouragements, ses critiques et ses précieux conseils qu'il m'a donnés et sans lesquels ce travail n'aurait pas été possible. Je tiens également à lui exprimer toute ma reconnaissance pour la confiance qu'il m'a toujours témoignée, pour m'avoir laissé une grande liberté d'action et pour m'avoir offert d'excellentes conditions pour mener à bien mes travaux de recherche. Je lui suis reconnaissante pour sa disponibilité, pour ses nombreux conseils et pour les innombrables heures de relectures. Merci par-dessus tout d'avoir guidé mes premiers pas dans le monde de la recherche scientifique.

Je tiens à remercier avec beaucoup de gratitude à madame Fatma ABDELHEDI, Directrice du laboratoire CBI^2 au sein de la société TRIMANE, qui m'a fait confiance et qui m'a mis dans les meilleures conditions pour réaliser cette thèse, pour m'avoir si bien accueillie au sein de son équipe afin que je puisse mener à bien cette thèse, pour la qualité de sa supervision, son soutien constant, ses conseils inlassables, ses encouragements continus tout au long du processus de recherche et de rédaction de cette thèse. Sa grande capacité d'écoute et sa grande patience m'ont permis de continuer à avancer. Je suis particulièrement admirative de sa constante bienveillance et de sa simplicité. J'espère humblement pouvoir imiter ces qualités dans ma future carrière.

Je remercie très sincèrement monsieur Omar BOUSSAID, Professeur à l'Université Lyon 2 et, monsieur Slimane HAMMOUDI, Professeur à l'ESEO d'Angers pour avoir accepté d'être rapporteur de ce mémoire, pour leurs remarques pertinentes et pour leur participation à mon jury de thèse.

Je tiens à remercier également madame Faten ATIGUI, Maître de conférences au CNAM de Paris et monsieur Elhadj BENKHELIFA, Full Professor à l'Université de Staffordshire (GB), pour tout l'intérêt qu'ils ont manifesté envers mon travail et pour l'honneur qu'ils m'accordent en participant au jury.

Je remercie les gestionnaires de la Faculté d'informatique mesdames Michèle CUESTA, Florence THERY et Lydie BALLABRIGA pour la disponibilité, l'aide généreuse et la gentillesse dont elles ont toujours fait preuve à mon égard.

J’adresse mes chaleureux et sincères remerciements à monsieur Lionel RIGAUD, directeur général de TRIMANE qui m’a fait confiance et qui m’a mis dans les meilleures dispositions pour réaliser cette thèse. Je le remercie également de participer au jury.

Je dis merci (par ordre alphabétique) à Bilel SDIRI, Elyes SAHLI, Mohamet GUYE, Naguib CHALAL, Stéphanie PESCHEUX, collègues à TRIMANE avec qui j'ai travaillé et cogité à travers d'innombrables séances de travail. De façon plus générale, je remercie tous les collègues à TRIMANE, avec qui et auprès de qui j'ai énormément appris au cours de ces trois ans de thèse.

Je remercie ma meilleure amie Asma pour son soutien inconditionnel tout au long de cette thèse. Sa présence encourageante, ses conseils précieux et son amitié sincère ont été des éléments clés de mon parcours académique.

Pour finir, je souhaiterais remercier toute ma famille qui n’a eu de cesse de me soutenir et de croire en moi tout au long de mes études ; notamment, mes chers parents Khalil et Hager, ma grand-mère Fatma et ma sœur Inès, à qui je dédie cette thèse. Je leur suis très reconnaissante de m’avoir appris à être ambitieuse et à ne jamais baisser les bras. Mes remerciements ne pourront jamais égaler leur assistance et leur amour qui m'a apporté du soutien au moment où j'avais besoin d'aide. C’est grâce à eux et pour eux que je suis qui je suis aujourd’hui. Je tiens particulièrement à exprimer ma profonde gratitude à ma mère Hager, la maman parfaite sur tous les plans qui a consacré toute sa vie pour notre bonheur. Merci pour les sacrifices qu’elle a dû faire pendant mes longues années d’études et pour le soutien indéfectible dont elle a fait preuve lors des moments difficiles que j’ai passés pour préparer ma thèse. Qu’elle sache que vivre loin d’elle était l’épreuve la plus difficile de ma vie.

Je dédie ce travail à l'âme de mon grand-père…

Sommaire

Table des figures

Liste des tableaux

Liste des sigles et acronymes

BD	Base de Données
BI	Business Intelligence
SI	Système d'Information
UML	Unified Modeling Language
XML	eXtensible Markup Language
ED	Entrepôt de Données
CSV	Comma Separated Values
QVT	Query/View/Transform
MD	Magasins de Données
LD	Lac de Données
SD	Système Décisionnel
ETL	Extract Transform Load
ELT	Extract Load Transform
MDA	Model Driven Architecture
PIM	Platform Independent Model
PSM	Platform Specific Model
CIM	Computation Independent Model
JSON	JavaScript Object Notation

INTRODUCTION GÉNÉRALE

L'avènement de l'analyse des données massives a répondu à l'essor du Big Data qui a débuté dans les années 1990. Bien avant que le terme « Big Data » ne soit inventé, ce concept était appliqué lorsque les entreprises utilisaient de grandes feuilles de calcul pour analyser les données numériques et rechercher des tendances. Ces dernières années ont vu l'explosion des données générées et stockées par un grand nombre de dispositifs informatiques associés aux systèmes d'information des entreprises, aux réseaux sociaux et aux objets connectés (Internet des objets). Selon le rapport « Data Age 2025 » publié par International Data Corp. [1], la quantité de données générées et collectées par les systèmes informatiques devrait atteindre 175 zettaoctets (ZB) d'ici 2025, soit cinq fois la quantité de données générées en 2018.

Les bases de données (BD) ainsi constituées sont désignées par l'expression « Big Data » et vérifient la règle dite des « 3V » : Volume, Variété et Vélocité. Les approches classiques de stockage des données telles que les entrepôts de données (ED) [2], [3] s'avèrent inadaptées pour gérer les mégadonnées, notamment en raison de leur hétérogénéité [4], [5]. Par conséquent, il existe un besoin pressant de développer des méthodes pour stocker, organiser, interroger et analyser efficacement les mégadonnées [6]. Dans ce contexte, le concept de « lac de données » (data lake) a été proposé par Dixon, définissant ainsi un référentiel centralisé qui stocke des données de différentes sources dans leur format d'origine [7]. Ils réunissent plusieurs sources de données de différents types et formats tels que des BD relationnelles, des BD NoSQL, des fichiers CSV, des fichiers textes, des feuilles de calcul. Les données peuvent être structurées, semi-structurées ou non structurées, et elles peuvent provenir de sources internes et externes. Ces données massives, complexes et hétérogènes représentent un réservoir de connaissances essentiel pour les acteurs-métier ; mais le volume des données ainsi que la diversité des structures et des formats constituent un obstacle majeur aux traitements décisionnels. Certains travaux de recherche mettent l'accent sur la nécessité d'un système de métadonnées efficace pour faciliter l'accès aux données contenues dans un lac de données (LD). L'absence de métadonnées pourrait transformer le LD en un « marécage de données » (*data swamp* en anglais) [8].

Les données médicales sont stockées sous différentes formes et dans des systèmes différents, ce qui rend leur intégration difficile. Cependant, exploiter pleinement les données massives nécessite de résoudre des problématiques importantes dans le domaine de la gestion des données complexes. Ces données, dont la gestion est à la charge des entreprises qui les ont collectées, doivent pouvoir être sélectionnées, transformées et réorganisées pour être analysées. Il est donc très difficile pour des acteurs-métier d'accéder avec plus de clarté aux données dont ils ont besoin et d'en tirer bénéfice pour transformer ces données en valeur.

Contexte industriel : la société TRIMANE

Cette thèse de doctorat s'inscrit dans le cadre d'une convention CIFRE entre le laboratoire IRIT[1] - Université Toulouse 1 Capitole et la société TRIMANE[2]. TRIMANE est une société de services, fondée en 2005, experte dans le domaine de Data et l'informatique décisionnelle (BI). TRIMANE accompagne ses clients dans la transformation et la valorisation de leurs données pour assurer une meilleure productivité et compétitivité. TRIMANE accompagne ses clients dans le domaine des données, des systèmes décisionnels et data driven dans divers secteurs notamment la santé, l'énergie, le droit, la finance et l'assurance.

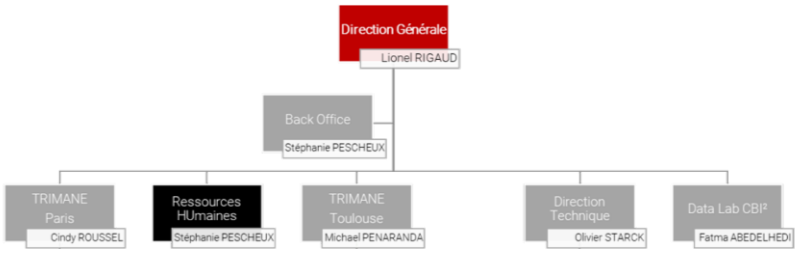


Figure 1 – Organigramme de TRIMANE

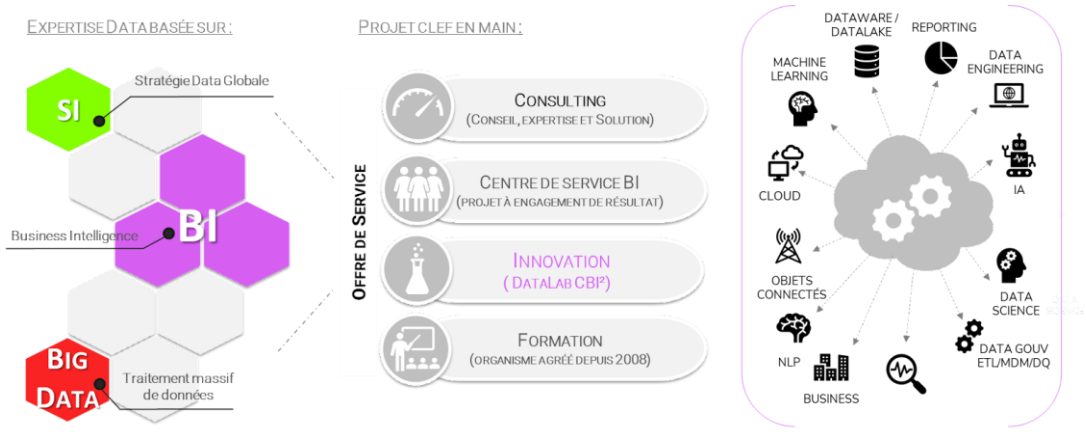


Figure 2 – Périmètre d'intervention

[1] https://www.irit.fr/
[2] https://trimane.fr/

TRIMANE possède son laboratoire de recherche CBI^2 (Centre d'expertise en Business Intelligence et Big Data). Le laboratoire CBI^2 vise à promouvoir la recherche et le développement des systèmes décisionnels innovants. Il veille, développe, teste et étudie les dernières évolutions technologiques du marché. Sous la direction de Dr Fatma ABDELHEDI, il permet de garantir un niveau de technicité élevé des équipes, intervenant sur des problématiques innovantes et complexes (Big Data, cycle de vie des données, systèmes NoSQL, Data Lake, Computer Vision, NFT, Data Driven, etc. Les travaux de cette thèse s'inscrivent dans le développement de CBI^2 pour incorporer de nouvelles solutions innovantes dans les offres commerciales de TRIMANE. CBI^2 travaille avec des clients issus de secteurs économiques différents. Parmi eux se trouvent les sociétés d'assurance maladie connues sous le nom de mutuelles de santé qui sont des organismes fournissant une couverture complémentaire pour les frais de santé non pris en charge par l'assurance maladie obligatoire. Ceci peut inclure des remboursements pour des médicaments, des consultations chez des spécialistes, des soins dentaires, des lunettes, des prothèses auditives, etc. Ces mutuelles de santé collectent et gèrent une grande quantité de données relatives à leurs membres et à leurs prestations. Ces données sont collectées et stockées dans un LD. Ces données volumineuses comprennent généralement des informations administratives, médicales et financières. L'accès à ces données est bénéfique pour les mutuelles de santé en leur permettant de prendre des décisions et d'optimiser leurs offres de services. Par exemple, en accédant puis en analysant les données médicales et les antécédents de santé des membres, les mutuelles de santé identifient les profils à risque élevé pour certaines maladies. Cela leur permet de mettre en place des programmes de prévention ciblés pour ces membres, tels que des campagnes de sensibilisation, des dépistages réguliers ou des incitations à adopter un mode de vie différent.

Ainsi, ces organismes extraient des connaissances à partir de données à l'aide de méthodes issues de la BI. La société TRIMANE souhaite accompagner ses clients dans l'accès aux données du LD. C'est cette problématique métier qui a motivé les travaux de recherche menés dans le cadre de notre thèse.

Problématique de recherche

Dans le cadre de cette thèse, nous avons répondu à un besoin métier spécifique des acteurs-métier du domaine médical pour l'exploitation d'un LD. Il s'agit de permettre à des professionnels de santé (personnels hospitaliers, de mutuelles, de cliniques) d'accéder à des BD volumineuses et diversifiées tant sur les types que sur les formats des données stockées. Cette application qui exploite un LD constitué de BD administratives et médicales, se heurte aux limites actuelles des logiciels de BI. En effet, des outils commerciaux permettent d'extraire des

données (Talend[3]) puis de les analyser par des acteurs-métier (Tableau[4] et Business Object[5]). Mais la complexité des structures de données présentes dans le LD constitue un obstacle majeur à l'accessibilité des données par des acteurs-métier. Ainsi, les particularités de notre application imposent de nouvelles contraintes liées à la variété des sources qui composent le LD. Parmi ces contraintes, on peut citer les données stockées en format natif qui sont souvent inexploitables en l'état ainsi que la redondance de données dans des sources de données différentes (les mêmes patients décrits selon des points de vue spécifiques). A ceci s'ajoute la difficulté qu'éprouvent les acteurs-métier pour sélectionner des données utiles à leurs requêtes. Cette thèse se focalise sur deux problématiques complémentaires :

- regrouper, dans un espace commun, les sources extraites d'un lac de données

- rendre exploitable cet espace pour faciliter le requêtage par des acteurs-métier

Contributions et organisation du manuscrit

Pour répondre à cette double problématique, nous proposons d'abord d'intégrer les données dans un entrepôt NoSQL centralisé. Ce choix se justifie d'une part par la diversité des types de données contenus dans les sources et d'autre part par la capacité des systèmes NoSQL à représenter des objets de types et de formats variés. La transformation des sources vers l'entrepôt repose sur l'architecture normalisée MDA basée sur un développement dirigé par les modèles [9]. Il convient de noter que la problématique présentée dans cette thèse a été restreinte à l'ingestion des seules BD relationnelles et BD NoSQL orientées documents contenues dans le LD.

Ensuite, une fois l'entrepôt NoSQL élaboré, nous spécifions une solution pour permettre à des acteurs-métier non informaticiens d'exploiter les données correspondantes à leurs besoins. Nous proposons des mécanismes de personnalisation basés sur un historique des accès réalisés par les acteurs-métier sur le LD.

Afin de vérifier la faisabilité de notre solution, nous avons développé un prototype destiné à l'exploitation du LD médical. Nous avons montré également

[3] Talend https://www.talend.com
[4] Tableau Software https://www.tableau.com/
[5] Business Object https://www.sap.com/

la pertinence de nos propositions grâce à une validation réalisée sur une application TRIMANE.

Nous avons organisé le manuscrit en 5 chapitres :

Le chapitre 1 présente le contexte de nos travaux ainsi que notre problématique : l'élaboration d'un entrepôt à partir d'un LD et l'accès à ces données via l'élaboration assistée d'un magasin.

Dans le chapitre 2, nous étudions les principaux travaux liés aux problématiques citées. Deux aspects sont notamment abordés : les processus d'élaboration d'entrepôts NoSQL à partir de sources hétérogènes (les mécanismes de transfert des données) et les techniques de recommandation dans un cadre décisionnel.

Le chapitre 3 est consacré à nos propositions en matière d'élaboration d'un entrepôt NoSQL à partir d'un LD. Il décrit le processus DLtoDW.

Le chapitre 4 présente les mécanismes de personnalisation et de recommandation DWtoDM associés au processus d'élaboration de magasins. L'objectif visé est d'assister les acteurs-métier dans l'exploitation des données.

Le chapitre 5 détaille le prototype développé pour mettre en œuvre notre processus d'élaboration. Ce système, dénommé DLtoDM, prend en compte l'ensemble des mécanismes proposés précédemment.

Enfin, nous concluons ce mémoire en mettant en exergue les travaux réalisés. D'une part, nous rappelons les caractéristiques de nos contributions et réalisons un positionnement par rapport à l'état de l'art. D'autre part, nous abordons les principales perspectives de recherche envisagées pour améliorer les travaux dans lesquels s'inscrit cette thèse.

Les travaux de cette thèse on fait l'objet des publications suivantes : [9], [10], [11], [12] et [13].

CHAPITRE 1 CONTEXTE ET PROBLÉMATIQUE

Dans un monde de plus en plus numérisé, les organisations sont confrontées à une avalanche de données provenant de sources diverses telles que les systèmes transactionnels, les médias sociaux, les capteurs IoT, les applications mobiles, etc. Ces données massives, communément appelées Big Data, offrent un potentiel immense pour générer des connaissances utiles et soutenir la prise de décisions stratégiques. Les systèmes décisionnels jouent un rôle clé dans l'exploitation de ces données massives en fournissant les outils et les processus nécessaires pour collecter, gérer et analyser ces données à grande échelle. Les entreprises ont rapidement compris la valeur potentielle de ce « nouvel or noir » [14] pour leur prise de décision et leur avantage concurrentiel. Cependant, le traitement de ces données nécessite des outils et des technologies spécifiques pour stocker, gérer et analyser ces données à grande échelle. Les LD ont été développés pour répondre aux besoins croissants de stockage des données brutes et hétérogènes provenant de multiples sources. Contrairement aux ED qui exigent une organisation dédiée aux applications décisionnelles, les LD permettent de stocker les données telles qu'elles sont, sans transformation.

Cette thèse se propose de réunir des sources hétérogènes contenues dans un LD et de rendre exploitables ces données par des acteurs-métiers non informaticiens afin d'observer leurs activités et de transformer ces données en valeur.

Ce chapitre présente le contexte de nos travaux ainsi que la problématique traitée. Nous précisons trois concepts clés de notre étude : l'informatique décisionnelle, les données massives et les LD. Ensuite, nous présentons notre problématique et l'étude de cas qui a motivé nos travaux.

1.1. Contexte de nos travaux

1.1.1. Systèmes décisionnels

Les systèmes d'aide à la décision (aussi appelés systèmes décisionnels) ont fait leur apparition dans les années 1960, mais il a fallu attendre les années 1980 pour voir les premières tentatives de regrouper différentes sources de données dans un seul système. Les premiers outils d'informatique décisionnelle étaient souvent des systèmes propriétaires, coûteux et difficiles à utiliser, ce qui a limité leur généralisation dans l'industrie.

L'informatique décisionnelle, ou Business Intelligence (BI) en anglais, est un ensemble de méthodes, technologies et outils informatiques qui vise à collecter, intégrer les données brutes, les présenter et les analyser pour faciliter la prise de décision.

En 1970, E.F. Codd a introduit le modèle relationnel pour stocker les données de manière plus indépendante et efficace que les approches précédentes. Dès les années 1980, les systèmes de gestion de base de données relationnels (SGBDR) ont été largement adoptés par les entreprises pour leur facilité d'utilisation et leur efficacité dans les analyses de données. Cependant, la simplicité des structures relationnelles a constitué un obstacle pour effectuer des analyses complexes. Pour répondre à ce besoin, Codd et al. [15] ont proposé un modèle multidimensionnel pour permettre à des acteurs-métier (non informaticiens) de réaliser des analyses OLAP (On-Line Analytical Processing) sur des BD créée à cet effet. L'avènement de cette technologie dans les années 90 a entraîné l'émergence d'architectures décisionnelles au sein des entreprises pour favoriser la prise de décision [16], [17]. Dans ce document, nous adoptons la définition de [17] pour définir les architectures décisionnelles comme une partie du système d'information dédiée aux applications décisionnelles.

Les architectures décisionnelles sont généralement [18] composées de trois niveaux de stockage (source, ED et MD) et de deux niveaux fonctionnels (outils ETL et outils de restitution et d'analyse) comme le montre la figure 1.1. Nous définissons ci-après les principaux éléments qui apparaissent dans la figure.

Les sources de données peuvent être internes ou externes à l'entreprise. Les sources internes incluent entre autres les données transactionnelles (données financières, de ressources humaines, de marketing, etc.). Les sources externes peuvent inclure des données économiques, démographiques ou météorologiques, ainsi que des données provenant de fournisseurs tiers ou de médias sociaux. Les données peuvent également être structurées ou non structurées telles que des documents textuels, des images et des vidéos.

L'entrepôt de données (ED) est l'élément central de l'architecture décisionnelle. Selon Bill Inmon :

"Un entrepôt de données est une collection de données intégrées, orientées sujet, non volatiles, historisées, résumées et disponibles pour l'interrogation et l'analyse." [19]

Le concept de magasin de données (MD) est étroitement lié à celui d'ED. Selon la vision de KIMBALL, un MD peut être défini comme un sous-ensemble d'un ED dédié à des analyses spécifiques. Un MD répond souvent aux besoins d'un acteur-métier ou d'un département spécifique, c'est-à-dire d'une classe d'acteur-métiers.

Le processus ETL (Extract-Transform-Load) consiste à extraire les données des sources de données, à les traiter avant de les insérer dans l'ED ou le MD. Ces trois traitements peuvent être détaillés comme suit [18] :

1. **L'extraction ou la collecte des données :** il s'agit de l'extraction de données provenant d'une ou plusieurs sources. Quand les sources sont multiples, il convient d'homogénéiser les données pour les rendre compatibles et cohérentes.

2. **La transformation des données :** dans cette phase, les données sont nettoyées, synthétisées voire enrichies pour assurer leur qualité et leur cohérence.

3. **Le chargement des données :** les données sont chargées dans une base décisionnelle matérialisée qui peut être un ED ou un MD.

L'ED est généralement une base de données relationnelle dont les tables sont organisées en forme d'étoile (parfois en flocon ou en constellation). Il existe une table de faits au centre qui contient les données mesurées. Les tables de dimensions décrivent les données de la table de faits. Ces tables de dimensions permettent aux acteurs-métier d'analyser les données de l'ED.

Les outils d'analyse permettent aux acteurs-métier d'explorer et analyser les données stockées dans l'ED ou dans un MD. Ces outils sont conçus pour faciliter l'interrogation et l'exploration des données ainsi que la création de rapports et de tableaux de bord pour une présentation visuelle des résultats.

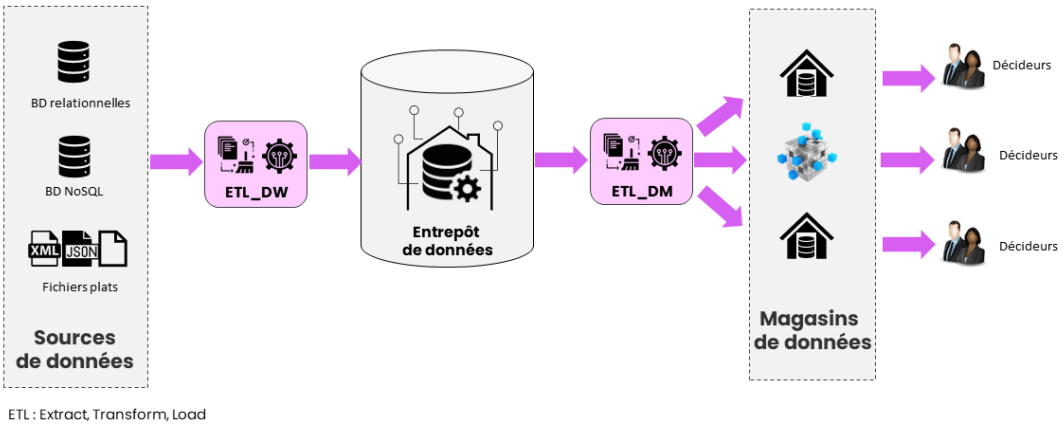


Figure 1.1 – Architecture décisionnelle

L'architecture décisionnelle peut être mise en œuvre selon deux ou trois niveaux de décomposition.

Architecture à deux niveaux : cette architecture distingue les sources de l'ED où les données sont organisées selon un modèle multidimensionnel. Les acteurs-métier ont un accès direct aux données de l'ED.

Architecture à trois niveaux : l'ED contient l'ensemble des données d'analyse ; il est organisé selon un modèle de bases de données standard, c'est-à-dire relationnel ou objet. Seuls des administrateurs de données (des informaticiens) gèrent l'ED ; les acteurs-métier n'ont pas un accès direct à ces données. Des MD sont extraits de cet ED et dédiés à des acteurs-métier. Selon les besoins d'analyse exprimés, les MD sont organisés selon des schémas appropriés à des besoins précis des utilisateurs. La figure 1.1 présente une architecture décisionnelle à trois niveaux (sources, ED et MD). Les MD sont destinés à l'analyse, ils doivent donc être réorganisés selon un modèle multidimensionnel.

Au début des années 2000, un nouveau type de BD est apparu. Il permet de stocker des données structurées mais aussi des objets hiérarchiques (documents) ainsi que des données peu ou pas structurées (texte, images, vidéo). Ainsi, les données destinées à l'analyse comprennent les réseaux sociaux (Facebook en 2004, Twitter en 2006, etc.), les informations issues des objets connectés (Internet des objets), ainsi que des courriels, des fichiers journaux (logs), des images, etc. [20]. Ces données massives sont communément appelées mégadonnées.

1.1.2. Mégadonnées

Ces données, appelées aussi Big Data en anglais, données massives ou mégadonnées, sont des données très volumineuses et complexes pour être traitées rapidement. Elles sont progressivement diffusées et installées à la fois dans les entreprises et dans notre vie de tous les jours. Ceci est dû d'une part à l'accroissement considérable de la puissance des ordinateurs et à leur diffusion dans nos environnements personnel et professionnel et d'autre part à la numérisation des activités qui s'est généralisée et a généré des quantités considérables de données de toutes sortes : photos, textes, courriels, etc. Ces données de tous types, qui se présentent sous des formats multiples, sont collectées, souvent stockées mais ne sont pas organisées en vue d'une utilisation particulière. Nous pouvons donner une définition simple et générale de ce type de données.

Les mégadonnées désignent des données volumineuses, variées, générées à grande vitesse et qui nécessitent des technologies et des techniques de stockage particulières.

Les mégadonnées sont caractérisées par les 3V : le Volume, la Variété et la Vélocité. Plus précisément, ces caractéristiques sont définies de la manière suivante [21], [22] :

- **Volume** : quantité de données stockées pouvant atteindre plusieurs téraoctets, pétaoctets ou même exaoctets de données.

- **Variété** : diversité de types et de formats de données ; par exemple, on peut trouver des données structurées (tables relationnelles, feuilles de calcul), des données non structurées (textes, images, audio) et semi-structurés tels que des documents XML ou JSON.
- **Vélocité** : vitesse à laquelle les données sont générées, collectées et traitées en temps réel ou quasi-temps réel.

En plus de ces trois caractéristiques communes, certains auteurs ont proposé d'autres caractéristiques des mégadonnées [23], généralement sous la forme de V. Nous en citerons seulement deux :

- **Véracité :** qualité et fiabilité des données pour garantir leur provenance, leur exactitude et leur cohérence.
- **Valeur :** extraction de la valeur ajoutée à partir des connaissances cachées de ces mines d'informations.

Face à des volumes de données massifs, à des besoins d'analyse en temps réel et à des transformations complexes, le processus classique d'ETL est devenu moins adapté [24]. En effet, l'ETL permet d'extraire les données sources puis de les transformer avant de les charger dans le système cible. Cependant, ce processus peut être lent et gourmand en ressources car il nécessite souvent de transférer et de transformer des quantités massives de données avant de les charger dans la BD cible. Le processus ELT (Extract-Load-Transform) est plus utilisé pour répondre à ces nouvelles exigences. Ce processus correspond aux LD, ce qui a permis aux organisations de charger d'abord les données brutes dans une BD distincte puis d'appliquer les transformations nécessaires en fonction des besoins d'analyse.

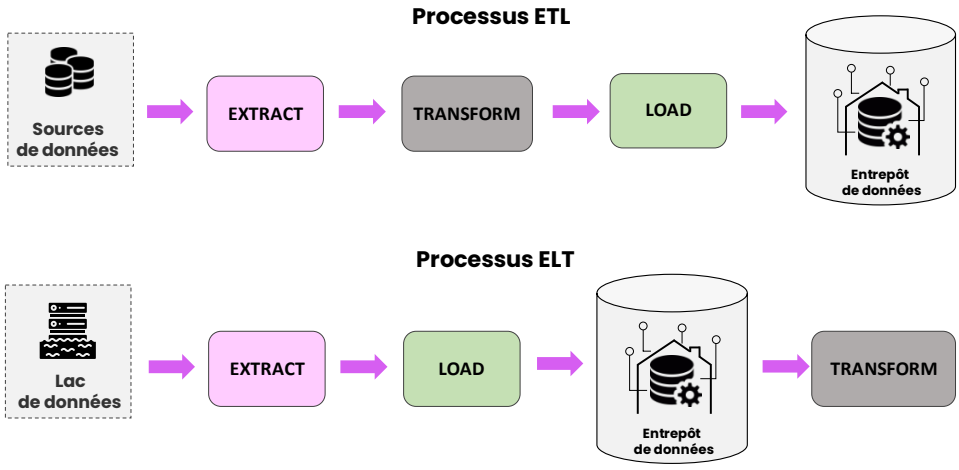


Figure 1.2 – Processus ETL et ELT

Les mégadonnées se heurtent aux limites des systèmes relationnels les plus répandus actuellement. Ces derniers assurent la cohérence et les transactions ACID. Une transaction est une unité de traitement qui maintient la cohérence des données en respectant les quatre propriétés suivantes :

- **Atomicité :** cette propriété assure que toutes les opérations composant une transaction sont traitées comme une unité indivisible. Cela signifie qu'une transaction est soit entièrement exécutée, soit complètement annulée en cas d'échec.
- **Cohérence :** la cohérence garantit que toutes les données dans la base de données respectent un ensemble de règles prédéfinies. Une transaction ne peut pas entraîner un état incohérent de la BD. Si une transaction viole ces règles, elle est annulée, préservant ainsi l'intégrité des données.
- **Isolation :** l'isolation assure que les transactions concurrentes ne se voient pas mutuellement jusqu'à ce qu'elles soient terminées. Ceci signifie qu'une transaction en cours ne peut pas accéder aux données d'une autre transaction en cours, évitant ainsi les conflits et les incohérences.
- **Durabilité :** la durabilité garantit que les modifications apportées par une transaction sont permanentes, même en cas de défaillance du système (panne de courant, blocage du système, etc.). Une fois qu'une transaction est validée, ses effets sont stockés de manière fiable dans la BD.

Aujourd'hui les systèmes NoSQL (Not-Only SQL) sont utilisés pour gérer les mégadonnées [25]. Ces systèmes répondent aux limites des BDR concernant le stockage des données semi-structurées et non structurées. Cependant, le maintien des propriétés ACID n'est généralement pas assuré dans ces nouveaux systèmes de gestion de BD. Cela est dû à l'absence d'un schéma de données prédéfini.

Les systèmes NoSQL ont été développés selon plusieurs approches : orientée clé-valeur, orientée-colonnes, orientée-documents et orientée-graphes. Il est impératif de choisir l'outil qui convient le mieux au problème à résoudre, tant du point de vue de la modélisation que de la distribution des données. Le théorème CAP énonce que dans un système distribué, il est impossible de garantir simultanément les trois propriétés suivantes [26] :

- **Consistency (C - Cohérence)** : tous les nœuds du système voient les mêmes données au même moment, quelles que soient les modifications apportées.

- **Availability (A - Disponibilité) :** les requêtes de lecture et d'écriture sont toujours satisfaites, assurant ainsi la disponibilité pour les opérations de lecture et d'écriture.
- **Partition tolerance (P - Tolérance aux pannes) :** la seule raison qui peut entraîner l'arrêt d'un système est la coupure totale du réseau. En d'autres termes, même en cas de dysfonctionnement d'une partie du réseau, le système continue de fonctionner.

Ainsi, pour concevoir une architecture distribuée, il est nécessaire de choisir deux de ces trois propriétés, ce qui donne lieu à trois conceptions possibles :

- **CP :** les données sont cohérentes entre tous les nœuds du système et ce dernier est tolérant aux pannes. Cependant, cela peut entraîner des problèmes de latence ou de disponibilité.
- **AP :** le système réagit de manière performante tout en étant tolérant aux pannes. Cependant, il n'y a aucune garantie de cohérence des données entre les nœuds.
- **CA :** les données restent cohérentes entre tous les nœuds tant que ceux-ci sont en ligne. Toutes les opérations de lecture/écriture des nœuds concernent les mêmes données. Cependant, en cas de problème réseau, certains nœuds peuvent se désynchroniser au niveau des données, ce qui entraîne une perte de cohérence.

1.1.3. Lacs de données

Le concept de LD est devenu de plus en plus répandu dans le domaine du stockage de gros volumes d'information. En effet, les organisations cherchent à bénéficier du formidable potentiel que représentent l'exploitation des données internes et externes aux entreprises pour prendre des décisions éclairées.

Un lac de données peut être défini comme un regroupement contextuel de sources hétérogènes, exploitées par ailleurs (feuilles de calcul, BD relationnelles, vidéo, etc.). Ces données sont généralement stockées dans leurs formats natifs sans se préoccuper, dans un premier temps, de l'exploitation qui en sera faite.

L'apport majeur du concept de LD se situe dans sa capacité à collecter et à conserver des données rapidement sous leur format d'origine. Ainsi, une des raisons majeures qui a conduit à la création de LD provient du besoin des entreprises pour analyser des flux de données en croissance exponentielle. En effet, ces données sont généralement produites sous des formes spécifiques et stockées dans des environnements indépendants. Un LD va

permettre de regrouper à la fois des flux de données générés par les systèmes de production et des ensembles de données existants et exploités par ailleurs (BD relationnelles, feuilles de calcul, documents semi-structurés, listes de valeurs, etc.) sans les transformer ou les unifier préalablement. Dans notre cas d'étude (voir section 1.2.1), le LD regroupe plusieurs BD ; celles-ci sont exploitées par des centres de santé indépendants et ne font pas l'objet d'une coordination forte. Or, certaines applications médicales nécessitent un regroupement de ces données à des fins statistiques et décisionnelles.

Les ED et les LD peuvent coexister dans le même environnement [27]. On peut envisager trois cas de figure :

1. Un LD peut servir de source à un ED. Dans ce cas, les données du LD sont utilisées pour alimenter l'ED.

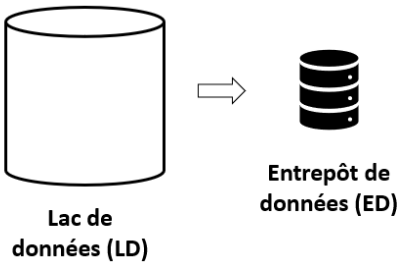


Figure 1.3 – Un LD source d'un ED

2. Un ED peut alimenter un LD. Dans ce cas, les données de l'ED sont migrées vers le LD où elles peuvent être stockées et analysées sans avoir besoin d'être transformées.

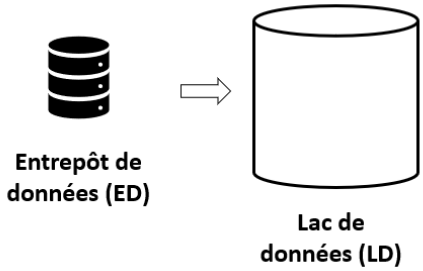


Figure 1.4 – Un ED source d'un LD

1.2. Cadre applicatif et problématique

La présente section vise à présenter la motivation et la problématique de notre recherche. Ces travaux trouvent leur raison d'être dans la convergence

de deux aspects fortement liés : un besoin métier bien identifié et des verrous scientifiques à lever pour exploiter les données.

1.2.1. Motivation

En informatique, les connaissances peuvent être représentées sous différents formats : du texte, des images ou des graphiques annotés, des bases de données relationnelles où la structure est explicite, ou des feuilles de calcul. Ces différents types de représentations de connaissances coexistent dans de nombreux environnements, notamment dans le domaine médical.

En 2019, le ministère français des solidarités et de la santé a défini une « feuille de route stratégique du numérique en santé ». Parmi les grandes orientations de ce plan, on trouve le renforcement de la gouvernance du numérique en santé et le déploiement des plateformes technologiques. Les objectifs sont de faciliter la coordination des soins et l'accès aux informations médicales, de renforcer la sécurité des données de santé, de développer des ressources et des services numériques pour assurer un accompagnement médical de la population ainsi que le traitement des données biométriques.

Les travaux de cette thèse sont motivés par un projet développé dans le domaine de la santé pour le compte d'un groupement de mutuelles santé. Ces sociétés d'assurance, issues de l'économie sociale et solidaire, proposent à leurs clients une prise en charge des frais médicaux qui vient en complément de ceux remboursés par un établissement public (la caisse d'assurance maladie).

Pour assurer la gestion de leurs clients, ces mutuelles santé sont confrontées à une augmentation importante des volumes de données à traiter. Ceci se vérifie d'autant plus que certaines de ces sociétés assurent le rôle de centre de sécurité sociale, c'est-à-dire que les clients transmettent la totalité d'un dossier de remboursement à une mutuelle qui effectue l'ensemble des traitements de données afférents au dossier. Pour suivre ces dossiers, de nombreux échanges sont nécessaires entre la caisse d'assurance maladie, les mutuelles et les professionnels de santé. Sous l'impulsion de l'Etat, la caisse d'assurance maladie a élaboré un Espace Numérique de Santé (ENS). Ce dernier permet de stocker les données médicales de chaque assuré, en croisant des données médico-administratives, des données hospitalières, des données de santé personnelles.

L'ENS doit permettre aux patients de consulter leurs informations de santé en ligne et de partager leurs données avec les professionnels de santé autorisés. Les professionnels de santé pourront quant à eux avoir accès à l'ensemble des informations médicales de leurs patients, et pourront les partager avec d'autres professionnels de santé pour améliorer la coordination des soins. Le projet ENS est composé de plusieurs volets,

comprenant notamment la mise en place d'un dossier médical partagé (DMP) pour chaque patient, d'un carnet de vaccination électronique, d'un système de messagerie sécurisé entre professionnels de santé, ainsi que d'un système d'identification unique pour chaque professionnel de santé.

Cette plateforme constitue un vaste réservoir de données qui répond à la définition d'un « Lac de données » puisque les informations sont stockées dans un format brut (non transformé) et ne sont pas organisées en vue de la prise de décision [28], [4]. Ce LD, issu de la réunion de plusieurs BD existantes ou en cours d'élaboration, se situe à la base de nos travaux. Les données publiques (pseudonymisées) qu'il contient, sont sélectionnées, nettoyées le cas échéant, agrégées et réorganisées dans des BD massives. Ces données sont de nature confidentielle mais le titulaire peut autoriser un professionnel ou un établissement de santé à consulter ou à alimenter son ENS. Tous les accès et actions effectués sur l'ENS par des non-titulaires sont tracés. Les mutuelles de santé ont donc un accès limité à l'ENS dont elles peuvent extraire des données dans le but de traiter les dossiers de leurs assurés et, plus largement, de réaliser des analyses de tout ordre (dans le respect des règles de confidentialité).

Pour chaque assuré, l'ENS contient notamment des données administratives, le dossier médical (constantes, archives d'imagerie médicale, comptes rendus, suivis thérapeutiques, etc.), l'historique des remboursements, des questionnaires. Lorsque l'ENS sera totalement déployé au niveau national, son volume sera considérable puisqu'il concerne 67 millions d'assurés. Compte tenu de la diversité et de la complexité des applications mises en œuvre, les utilisateurs de l'ENS ignorent généralement la nature et la structure des données que contient l'ENS et qu'ils souhaitent manipuler. D'où l'intérêt d'extraire des données, de les prétraiter et de les réorganiser dans un entrepôt ou des MD pour une utilisation ciblée.

Dans le cadre de ce projet, l'ENS constitue un véritable LD en raison (1) de la diversité des types, des supports et des formats de données (2) des volumes stockés pouvant atteindre plusieurs téraoctets et (3) de la nature brute des données dans le sens où les données regroupées ne sont pas unifiées. L'objectif du projet est d'étudier les mécanismes d'extraction des données de l'ENS et de leur organisation pour faciliter les analyses (Big Data Analytics). Il convient de souligner que, si l'ENS peut bien être considéré comme un LD, il ne contient pas toutes les caractéristiques que l'on peut trouver dans d'autres systèmes. Ainsi, l'ENS regroupe des ensembles de données hétérogènes mais il ne contient pas de données brutes au sens premier du terme. En effet, les données brutes ne sont généralement pas associées à des métadonnées permettant de leur donner une signification, celles-ci étant stockées par ailleurs. Dans le cas de l'ENS, les BD regroupées dans le LD sont hétérogènes mais chacune dispose de son propre schéma (séparé ou inclus dans les données pour le « schemaless »).

1.2.2. Problématique de recherche

Bien que d'importants efforts de recherche soient consacrés aux applications dans le domaine de la santé, les systèmes d'information médicaux représentent un domaine en pleine gestation [29] ; ils doivent être capables d'informer, de coordonner, de partager et de décider dans le cadre extrêmement sensible de la santé.

Les travaux menés dans le cadre de cette thèse CIFRE, répondent à deux enjeux majeurs.

Tout d'abord, les données médicales doivent être exploitées par des acteurs-métier. Il s'agit de permettre à des professionnels de santé (personnels hospitaliers, de mutuelles, de cliniques) d'avoir un accès assisté et facilité à des bases de données volumineuses et diversifiées tant sur les types que sur les formats des données stockées. Sollicitée par ses clients, la société TRIMANE souhaite développer des composants logiciels répondant à cette préoccupation et pouvant s'intégrer dans les systèmes d'information ouverts des entreprises clientes.

Le second enjeu est d'ordre scientifique lié à un environnement médical. Il vise à améliorer de façon significative l'accompagnement des usagers de l'Assurance Maladie en apportant de nouvelles démarches et de nouveaux outils aux personnels de santé et aux usagers eux-mêmes. Certes, il s'agit d'une évolution démarrée depuis plusieurs années dans le monde [30] ; mais elle devrait connaître une accélération majeure grâce à la généralisation du numérique ainsi qu'à l'utilisation des techniques de l'intelligence artificielle. Cette « révolution » numérique concerne aussi bien la prise en charge du patient lui-même que le suivi de l'environnement de santé et les prises de décision concernant le collectif ; pour prendre un exemple d'actualité, on peut citer l'étude de scénarios pour une réorganisation des ressources médicales face à une pandémie.

Par conséquent, l'utilisation de LD à des fins d'analyse décisionnelle soulève deux verrous dans notre cas d'étude.

Le premier verrou consiste à extraire des données à partir d'un LD existant puis à stocker ces données dans un entrepôt en vue d'y accéder. L'hétérogénéité des systèmes de stockage associé à la diversité des contenus du LD constitue un obstacle majeur à l'exploitation décisionnelle des données. Il convient donc de réorganiser ces données sous une forme unifiée, en prenant en charge la fusion des données "similaires" ainsi que la conversion des liens sous la forme de références entre objets. Or, peu de mécanismes sont offerts par les plateformes de stockage actuelles pour prendre en compte des BD hétérogènes tout en garantissant la cohérence des données et, par conséquent, leur qualité.

Le second verrou concerne l'accès aux données de l'entrepôt par des acteurs-métier. Dans notre environnement applicatif, le décideur-type est un non informaticien qui maîtrise bien son besoin-métier mais qui est peu rompu à l'élaboration de requêtes d'accès complexes, fussent-elles en langage SQL. En effet, l'entrepôt, même s'il a été élaboré à des fins décisionnelles, contient des mégadonnées dont les structures sont complexes. Il convient donc de lui offrir des mécanismes simples lui permettant de formuler son besoin de données sous forme de requêtes.

Il convient de souligner que nous limitons le champ de notre étude à des ensemble de données (fichiers) présentant les caractéristiques particulières. Ainsi, le LD ENS constitue l'entrée de notre processus et contient des données massives structurées et non structurées correspondantes à des BD relationnelles et des BD NoSQL orientées-documents. En effet, ces catégories de données représentent une partie importante des données de l'ENS sans toutefois couvrir la totalité de l'ENS.

Les BD NoSQL offrent deux principaux mécanismes pour gérer les relations entre les données : l'imbrication et les liens par référence. L'imbrication permet d'inclure des données directement à l'intérieur d'autres documents, formant ainsi une structure hiérarchique. Dans le cas des liens par référence (Exemple *DBRef* pour le système MongoDB[6]), un document fait référence à un autre document grâce à un identifiant. Contrairement aux systèmes relationnels, MongoDB ne met pas en œuvre les contraintes d'intégrité référentielle. En d'autres termes, il ne garantit pas automatiquement que les références pointent vers des documents existants. Cependant, certains travaux ont proposé des algorithmes pour garantir la cohérence des données dans le cas de MongoDB [31].

Nous supposons que les BD contenues dans le LD sont cohérentes puisqu'elles sont exploitées par des applications dans les centres de santé. L'ENS contient aussi des métadonnées spécifiques qui ont été ajoutées lors de sa création. Ainsi, chaque source, qu'elle soit relationnelle ou NoSQL, est décrite par un schéma. Par exemple, le schéma d'une BD relationnelle est extrait du SGBD qui la gère. Pour les BD NoSQL "schemaless", le schéma est produit grâce à une application particulière. Dans le cas de l'ENS, nous avons utilisé un logiciel mis au point par notre équipe [32] qui permet d'extraire un schéma logique d'une BD NoSQL orientée-documents. Dans la présentation de nos travaux, nous préciserons la nature et la forme de ces métadonnées.

L'entrepôt et les magasins générés par notre processus sont des BD NoSQL orientée documents ; à ce stade de notre étude, ils ne sont pas organisés selon un modèle multidimensionnel. De plus et bien que MongoDB soit le système NoSQL le plus utilisé actuellement dans l'industrie [33], nous avons

6 https://www.mongodb.com

choisi OrientDB[7] pour gérer l'entrepôt et les magasin . En effet, ce système NoSQL offre des fonctionnalités avancées, notamment la possibilité d'exprimer plusieurs types de liens sémantiques. Il est proche des concepts définis par l'ODMG[8] pour les systèmes objet, notamment en matière de structuration des données et de relations ; il offre en plus la propriété « schemaless ». Il sera ainsi plus aisé de restructurer les données de l'entrepôt pour faciliter les traitements décisionnels. En outre, aucun détail sur les techniques de traitement des données n'a été fourni.

7 https://orientdb.org/
8 http://www.odbms.org/odmg-standard/

CHAPITRE 2 ÉTAT DE L'ART

Les entreprises placent la capitalisation de leurs données au centre de leur démarche de transformation numérique dans le but de les exploiter ultérieurement. Les LD émergent comme l'une des solutions permettant de maximiser la valeur des données au sein des organisations. Ces systèmes sont spécifiquement conçus pour collecter, stocker et exploiter d'importantes quantités de données disponibles. Initialement considérés principalement comme des plateformes de stockage économique, ils ont évolué pour devenir un enjeu stratégique majeur dans la valorisation des données stockées pour les entreprises.

Ce chapitre présente l'état de l'art sur l'exploitation des données d'un LD. Ceci revient à permettre aux acteurs-métier d'accéder facilement aux données souhaitées. Ainsi ce chapitre est articulé autour de deux volets principaux :

- créer un entrepôt pour regrouper des sources hétérogènes
- assister l'acteur-métier dans le requêtage de ces données

Concernant le premier volet, la section 2.1 présente les travaux qui ont porté sur la gestion des données ingérées dans un LD. En ce qui concerne le deuxième volet, la section 2.2 présente un état de l'art général des solutions facilitant l'accès aux données pour la prise de décision.

2.1. Gestion des données ingérées dans les lacs de données

Dans ce qui suit, nous étudions les travaux liés à la gestion des données ingérées au sein d'un LD pour les rendre accessibles et exploitables. Certaines recherches ont été menées sur les architectures des LD c'est-à-dire la structuration et l'organisation des données ; d'autres ont considéré les métadonnées lors des manipulations des LD.

2.1.1. Organisation en zones

L'adoption des LD a connu une croissance significative dans l'industrie et la recherche depuis 2016. Les LD sont conçus pour stocker de grandes quantités de données quel que soit leur format de manière rapide et flexible. Ils ne se limitent pas à une seule technologie de stockage telle que HDFS (Hadoop Distributed File System) mais offrent un nouvel « écosystème » de données qui permet de croiser rapidement et facilement des données provenant de différentes sources. Dans ce sens, plusieurs travaux se sont

concentrés sur la proposition d'architectures de LD pour stocker, organiser et exploiter les données.

Des travaux considèrent le LD comme un espace de stockage pour des données massives. Cette architecture appelée "mono-zone" a été proposée par Capgemini[9] dans un document technique [34]. Elle permet aux utilisateurs de collecter et stocker les données dans leurs formats bruts. Elle se compose d'une unique zone de stockage où toutes les données sont ingérées dans leurs formats d'origine. Ces données peuvent provenir de différentes sources telles que des BD relationnelles, des systèmes de fichiers, des capteurs et des applications. Cette architecture offre l'avantage d'autoriser l'ingestion de données hétérogènes et volumineuses à faible coût. Cependant, elle peut transformer le LD en un « marécage de données » (*Data Swamp* en anglais) ce qui rend les données inexploitables.

Pour contourner ce problème, Bill Inmon [35] a publié un livre sur l'architecture des LD. Son objectif est d'éviter le stockage de données inutiles et inexploitables. Il préconise une transition des architectures des LD vers les modèles décisionnels ; cette solution permet de ne pas stocker uniquement des données brutes mais plutôt des données prétraitées ou "raffinées" à l'aide de processus de type ETL (Extract-Transform-Load). Inmon met en avant l'importance de l'utilisation des métadonnées, discute du profil des utilisateurs des LD (principalement les data scientists) et propose une organisation des données à l'intérieur du LD en fonction de leur type notamment les catégories telles que « analog data », « application data » et « textual data ». Son principal axe de réflexion se concentre sur cette catégorisation des données à intégrer dans le LD. Il est important de noter que l'auteur n'a pas abordé les aspects liés aux modalités de stockage ou au traitement des données dans cet ouvrage.

D'autres travaux organisent des données en zones selon leur degré de traitement c'est-à-dire leur niveau de maturité [36]. Par exemple, l'architecture AWS Lake[10] d'Amazon se compose de plusieurs zones : la zone d'ingestion, la zone de stockage, la zone de traitement et la zone de gouvernance et de sécurité [37]. Les données sont extraites des sources et chargées dans la zone d'ingestion. Ces données brutes sont ensuite stockées dans la zone de stockage. Au besoin, elles sont transformées au niveau de la zone de traitement. La zone de gouvernance assure la qualité, la gestion des métadonnées et le cycle de vie des données. Cependant, bien que les données brutes soient ingérées et conservées dans un référentiel de données central, les résultats des processus de transformation ne sont pas enregistrés dans ce même référentiel. Les données qui ont été nettoyées et transformées sont directement mises à la disposition des utilisateurs et ne sont pas stockées

[9] https://www.capgemini.com
[10] https://aws.amazon.com/fr/solutions/implementations/data-lake-solution/

dans un référentiel central. L'architecture du LD proposée dans [5] va au-delà de celle d'AWS Lake et adopte une architecture à sept zones. Les données ingérées sont stockées dans la zone de données brutes. Ces données sont transformées au niveau de la zone de raffinement en fonction des besoins (nettoyage, agrégation, normalisation ...). Les données transformées sont transférées dans la zone des données fiables. La zone « bac à sable » est destinée aux data scientists pour leur permettre de mener des analyses exploratoires et prédictives sur les données fiables. Vient ensuite la zone de consommation contenant les données qui sont régulièrement manipulées par des utilisateurs. La zone de gouvernance représente la dernière zone de cette architecture et communique avec les six autres zones dans le but de gérer le cycle de vie de données et leur qualité. Cependant, les auteurs ne mentionnent pas de manière explicite les méthodes de traitement des données applicables aux différentes zones.

2.1.2. Utilisation des métadonnées

Les métadonnées sont des informations fournies par les utilisateurs ou capturées directement par le système lors de l'ingestion et le traitement des données. Une donnée ingérée dans le LD est associée à un identifiant et annotée par un ensemble de métadonnées. Celles-ci peuvent décrire le contenu des données, leur format, leur origine et leur utilisation [38]. Elles peuvent également décrire les relations entre les données. Dans le cas d'un LD et en l'absence d'un schéma de données prédéfini, les métadonnées jouent un rôle essentiel en fournissant des informations contextuelles et des descriptions sur les données, empêchant ainsi le LD de se transformer en « marécage de données » (*data swamp*), c'est-à-dire une masse de données inutilisables. En effet, les métadonnées aident les utilisateurs à comprendre ce que contiennent les données, comment elles sont organisées, quelles sont leurs sources, et quelles sont les transformations dont elles ont fait l'objet.

Ainsi, plusieurs travaux ont proposé des méthodes d'extraction de métadonnées d'un LD. L'extraction de métadonnées est le processus de découverte des informations sur un ensemble de données, généralement les métadonnées structurelles sont extraites en premier lieu, mais aussi les informations sémantiques et les relations avec d'autres ensembles de données sont extraites.

Les métadonnées peuvent être générées de diverses manières. Une première approche implique la saisie manuelle d'informations par un utilisateur (ou le créateur) du LD [39]. Bien que cette méthode soit lente et laborieuse, elle s'avère nécessaire pour certaines métadonnées telles que les descriptions ou les catégorisations métier qui seraient difficiles à obtenir autrement.

Une méthode semi-automatique consiste à extraire automatiquement des informations à l'aide d'algorithmes qui analysent les données. Selon leur

nature, ces métadonnées extraites automatiquement peuvent nécessiter une intervention humaine pour validation ou correction [40].

Une méthode automatique permet l'extraction de métadonnées en fonction du type de données. En ce qui concerne les données structurées, l'article de Farrugia et al. [41] propose une méthode basée sur les schémas des BD relationnelles. Ensuite, les auteurs établissent une connexion entre deux tables si elles partagent au moins une colonne commune. Cependant, les critères exacts permettant de déterminer si deux tables ont une colonne en commun ne sont pas explicitement précisés. Nous supposons donc que la comparaison est basée exclusivement sur le nom de colonne et son type.

Guidice et al. [42] ont consacré leurs travaux à l'extraction de métadonnées relatives aux données non structurées (texte, image, vidéo, etc.) au sein des LD. Leur méthode se base sur la proximité des mots-clés et repose sur la théorie des graphes. Tout d'abord, chaque mot-clé est associé à un nœud distinct ; ces nœuds sont liés à un nœud central qui représente le document non structuré, tel qu'une vidéo. Ensuite, en utilisant un thésaurus comme BabelNet[11], les mots-clés ayant une relation selon ce thésaurus sont liés par une arête. Enfin, une analyse de similarité textuelle est effectuée entre chaque paire de mots-clés : si la similarité dépasse un seuil défini, une autre arête est créée pour relier les nœuds correspondants. Toutes ces informations sont ensuite enregistrées dans le système de métadonnées.

Diamantini et al. [43] ont proposé une approche en deux étapes qui peut être appliquée à des données hétérogènes (BD relationnelles, fichiers XML et vidéos). La première étape implique l'extraction de mots-clés à partir des données en utilisant des méthodes telles que l'extraction des intitulés d'attributs pour les données structurées ou l'identification des entités nommées pour les documents textuels. Ensuite, ces mots-clés sont comparés en utilisant soit une ontologie soit la distance de similarité N-gramme [44] puis ils sont liés entre eux s'ils présentent une forte similarité à la fois du point de vue syntaxique et sémantique. Ces métadonnées permettent de pallier l'absence d'un schéma pour faciliter la recherche, la découverte et la gestion des données au sein du LD.

Une fois les métadonnées extraites, elles doivent être réorganisées au sein d'un système de gestion de métadonnées pour permettre l'exploitation des données d'un LD. Ainsi, Quix et al. [45] présentent un système de gestion des métadonnées pour les LD appelée GEMMS. Ce système se compose de deux entités abstraites : les fichiers et les unités. Un fichier représente une source de données, tandis qu'une unité de données représente un élément de données identifiable au sein d'une source de données. Chaque fichier de données est constitué d'un ensemble d'unités de données, par exemple, un

[11] https://babelnet.org/

tableur peut être composé de plusieurs feuilles de calcul. Les fichiers de données et les unités de données peuvent être enrichis avec des métadonnées. Dans [46], GEMMS a été étendu pour représenter des schémas spécifiques pour les tables relationnelles, les fichiers JSON et les graphes de propriétés étiquetés de Neo4j.

Certains auteurs suggèrent de ne pas se limiter à décrire uniquement la structure des données intégrées dans un LD mais également d'englober la signification sémantique des données en recourant à des ontologies et à des vocabulaires propres au domaine. Ainsi, les travaux de Diamantini et al. [43] proposent une architecture où les données sont regroupées en fonction de leur signification. L'unité de base est l'objet qui peut être défini de différentes manières selon le type de données sources. Par exemple, dans une BD relationnelle, un objet correspond à une table et ses attributs, tandis que dans un document XML ou JSON, un objet représente une structure simple ou complexe ainsi que ses attributs. Les objets peuvent être liés entre eux par des liens de similarité où chaque lien possède un poids. Ces liens permettent de détecter les relations sémantiques entre les objets ; cela est utile pour les analyses et les visualisations de données. Les métadonnées sémantiques associées à un objet peuvent être utilisées pour enrichir sa signification. Par exemple, on peut associer à un objet des termes provenant d'une ontologie qui permettent de décrire sa sémantique de manière plus précise. Cette architecture facilite l'exploration et l'analyse des données car elle regroupe les données par thème ou par concept.

De même, Ansari et al. [47] présentent une méthode de profilage sémantique adaptée pour les LD. Le profilage sémantique est un processus qui consiste à identifier et à décrire la signification des données. Cette méthode permet d'améliorer l'exploration des données dans les LD en rendant les données plus compréhensibles et exploitables pour les utilisateurs. Le profilage sémantique repose sur l'utilisation de métadonnées portant sur la signification des données. Elles peuvent être collectées à partir de diverses sources telles que les descriptions de données, les vocabulaires et les ontologies. Le processus de profilage sémantique est effectué en deux étapes. Dans un premier temps, des métadonnées sont identifiées et collectées. Dans un second temps, ces métadonnées sont analysées pour identifier les relations entre les données.

Ces différents travaux mettent en évidence l'importance de considérer la gestion des métadonnées lors de la conception des LD. Il n'existe toutefois pas de consensus en ce qui concerne l'organisation des métadonnées ; c'est pourquoi les propositions sont assez variées : certaines sont des implantations de LD peu détaillées et dédiées à un cas d'usage spécifique, tandis que d'autres sont des modèles de métadonnées, plus explicites, mais manquant de généricité pour s'adapter à des types de données variés.

2.1.3. Centralisation des données

En l'absence d'un système multimodal capable de gérer plusieurs ensembles de données de manière uniforme, une solution consiste à centraliser ces données après les avoir uniformisées. Actuellement, le modèle objet correspond au modèle de données le plus complet et le plus général [48], c'est-à-dire capable de décrire tous types de structures de données. La centralisation des données peut ainsi reposer sur la transformation des ensembles de données hétérogènes vers une BD NoSQL unique basée sur un modèle objet. Des travaux de recherche ont étudié les transformations de données entre différents types de BD. Dans le cas du passage d'une BD relationnelle vers une BD NoSQL, l'un des points importants à prendre en considération est la conversion des liens ainsi que le respect des contraintes d'intégrité référentielle [49], [50]. Des études sur la migration d'une BD relationnelle vers une BD NoSQL ont proposé des mécanismes de transformation, automatiques ou semi-automatiques, pour convertir les modèles puis transférer les données [51], [52], [53]. Ainsi, l'article [54] propose un environnement logiciel générique pour migrer une BD relationnelle vers une BD NoSQL. Cette dernière peut suivre le modèle document, clé-valeur ou colonne. Le mécanisme de correspondance entre les concepts est basé sur l'utilisation de graphes. Dans le cas d'une BD NoSQL orientée-documents, chaque table correspond à une collection, chaque tuple à un document et chaque colonne à un champ. Les clés primaires deviennent des champs et les clés étrangères sont converties sous la forme d'imbrications de documents. Le système permet aux utilisateurs de personnaliser des règles de transformation pour prendre en compte certaines spécificités, notamment dans le schéma cible ou dans les données source. Une fois la correspondance établie entre les concepts, le système génère le code (Python ou JavaScript) permettant de transférer les données relationnelles dans la BD NoSQL. Une autre étude [55] a traité le transfert de données du SGBD MySQL vers le système MongoDB. Les clés étrangères de la BD relationnelle sont transformées en imbrications dans la BD NoSQL. L'algorithme de transformation a été implanté avec l'outil de développement Java NetBeans IDE dédié aux applications interactives. Ensuite, le processus génère automatiquement des fichiers texte au format Pentaho Data Integration [56] ; ces fichiers sont chargés dans un second temps dans MongoDB depuis MySQL. D'autres travaux ont étudié l'utilisation des références pour convertir les liens relationnels, c'est-à-dire transformer les clés étrangères par des pointeurs entre objets. C'est le cas dans l'article [57] qui présente une méthode de conversion d'une BD relationnelle en une BD NoSQL mais se concentre sur la conversion des liens lors de la migration vers MongoDB. Les auteurs indiquent que leur système est capable de sélectionner la méthode optimale pour convertir les liens en fonction des différentes cardinalités (one-to-one, one-to-many et many-to-many). Cette méthode reposerait sur l'utilisation d'un réseau de neurones et pourrait lier les documents en conséquence (par imbrication ou grâce à des références).

Cependant le fonctionnement de ces différents modules n'a pas été détaillé. De même, l'article [51] a étudié la transformation d'une BD relationnelle en une BD NoSQL (orientée-colonnes et orientée-documents). Les cardinalités permettent de choisir la forme des liens adaptée dans la BD NoSQL : l'imbrication de documents (dans le cas de liens un à un et un à plusieurs) ou l'utilisation des références (dans le cas de liens many-to-many). Les mécanismes proposés utilisent des métadonnées stockées dans des tables système associées à la BD relationnelle. Ils ont été testés sur des BD de petite taille mais le passage à l'échelle semble possible. Les auteurs de [58] ont proposé NoSQLayer, un processus automatique qui permet de migrer une BD relationnelle vers une BD NoSQL. Deux SGBD ont été utilisés dans ce travail : MySQL pour la source et MongoDB pour la cible. NoSQLayer se compose de deux modules : le module de migration de données et le module de mappage de données. Le module de migration de données est responsable d'identifier automatiquement tous les éléments de la BD source (par exemple, tables, attributs, relations, index, etc.), de créer une structure équivalente en utilisant le modèle de données NoSQL, et enfin, de migrer complètement les données. Ainsi, une table est transformée en une collection. Les enregistrements de chaque table sont transformés en documents où chaque attribut des tables est représenté par des champs dans ces documents. De plus, les relations existantes entre les tables sont représentées par des références de documents. NoSQLayer crée également une collection appelée « Metadata » pour stocker toutes les informations collectées lors de l'étape précédente, telles que les noms de tables, les noms d'attributs, les types d'attributs et les contraintes d'intégrité. Le module de mappage utilise cette collection de métadonnées pour vérifier si les contraintes d'intégrité sont respectées. Dans l'article [59], les auteurs ont présenté un outil pour la transformation des BD relationnelles vers plusieurs familles de BD NoSQL, à savoir les BD clé-valeur, colonnes, documents et graphes. Des algorithmes ont été proposés pour effectuer la transformation en NoSQL et obtenir une BD cible cohérente. Par exemple, pour transformer une BD relationnelle en une BD NoSQL orientée-documents de type MongoDB, une table est convertie en une collection et une colonne en un champ. Selon le type de lien, une clé étrangère devient une structure *DBRef* ou une imbrication de documents. De même, Erraji et al. [60] se concentrent sur le développement d'un processus ETL adaptés aux BD NoSQL. La première étape consiste à extraire les données à partir des BD relationnelles. Cette étape nécessite une modélisation de la structure de la source et de la cible. Une suite de règles a permis d'assurer le passage d'une BD relationnelle vers une BD MongoDB et d'adapter les schémas de la source et la cible. Ces règles transforment une table en une collection, une ligne en un document. Les auteurs ont considéré la transformation des liens d'association uniquement et utilisent l'imbrication des documents ou les références entre les documents selon les cardinalités. Puis le chargement des données dans les BD NoSQL cibles est effectué.

D'autres travaux ont utilisé des techniques de transformations de modèles comme MDA pour réaliser la migration des données d'une BD relationnelle vers des BD NoSQL [61]. Rappelons que l'architecture MDA propose une architecture de développement de logiciels à partir de modèles et de règles de transformation. Ainsi, l'article [62] propose des règles de transformation pour migrer d'une BD relationnelle vers une BD NoSQL en utilisant MDA pour la transformation automatique des modèles. Ainsi, chaque table est transformée en une collection, chaque ligne en un document et chaque colonne en un couple clé-valeur. Les liens (clés étrangères) sont gérés par imbrication de documents ou par référence. Ces règles de transformation ont été développées en utilisant les standards MOF 2.0 QVT[12] et Acceleo[13] afin d'accélérer et de faciliter la création de BD NoSQL sur le système MongoDB.

Le transfert des données depuis une BD NoSQL vers une autre BD NoSQL a également été étudié. Ainsi, dans l'article [63], des algorithmes ont été proposés pour convertir une BD NoSQL d'un modèle vers un autre ; par exemple du modèle clé-valeur vers l'orienté-document. Les relations entre données sont exprimées sous la forme de propriétés imbriquées ou par référence. Par ailleurs, les auteurs de [64] proposent un processus pour intégrer des données massives issues de médias sociaux (Facebook, Twitter et Youtube) dans une BD NoSQL. Un ensemble de règles de correspondance assure le transfert des données selon les deux modèles des systèmes NoSQL : orienté colonnes ou orienté documents. Ces règles sont complétées par un outil ETL (Talend) pour transférer ces données dans l'entrepôt NoSQL.

L'examen des contributions portant sur l'exploitation d'un LD montre que les travaux de recherche et développement actuels n'ont pas encore permis de mettre au point un système capable de manipuler un LD contenant des sources hétérogènes. Certains travaux ont étudié le passage d'une source vers une autre mais aucun ne prend en compte la transformation des différentes sources pouvant constituer un LD et permettant une manipulation unifiée. D'autre part, les techniques proposées ne s'adressent généralement pas à des acteurs-métier. Pour répondre à notre problématique, il sera donc nécessaire de proposer des mécanismes applicables à un LD existant et permettant à des acteurs-métier de cibler les données correspondantes à leurs besoins.

2.2. Accès aux données

Dans notre cas d'étude, les acteurs-métier accèdent à des données présentant des structures complexes. Il est donc difficile de proposer à ce type d'acteur-

[12] OMG, Q. (2011). Meta Object Facility (MOF) 2.0 Query/View/Transformation, v1.1.
[13] https://wiki.eclipse.org/Acceleo/Getting_Started

métiers, généralement des non informaticiens, un langage de type SQL pour interroger les données.

Parmi les outils facilitant l'accès aux données, des systèmes de recommandation (SR) ont été développés pour assister les acteur-métiers dans leurs recherches d'information.

Ces systèmes fournissent des recommandations personnalisées basées sur leurs préférences, leurs comportements passés ou d'autres informations contextuelles [65]. Les SR fonctionnent en utilisant différentes techniques d'apprentissage automatique et d'analyse de données pour capturer les préférences et les comportements des utilisateurs, ainsi que les caractéristiques des éléments recommandables [66]. Ces techniques comprennent :

- le filtrage collaboratif qui identifie les similarités entre utilisateurs [67] en se basant sur leurs historiques de préférences. Ensuite, il recommande à l'utilisateur les éléments appréciés par les utilisateurs similaires.
- le filtrage basé sur le contenu qui utilise des caractéristiques des éléments pour proposer des recommandations similaires à ceux déjà choisis par l'utilisateur [68].

Les SR ont évolué pour devenir un domaine de recherche bien établi, avec des applications dans de nombreux domaines [69], [70].

Dans les années récentes, des recherches sur la recommandation ont été menées dans les domaines des systèmes d'information, de la recherche d'information et des BD. Plusieurs travaux ont abordé la recommandation dans les ED [71]. Dans la littérature sur les systèmes de recommandation appliqués au décisionnel et à l'exploration et l'exploitation des BD, les travaux s'intéressent plus particulièrement à la recommandation des requêtes [72]. Celle-ci consiste à suggérer des requêtes en fonction de ses intérêts et/ou de son analyse en cours. Dans le cas d'une BD décisionnelle, la recommandation guide l'utilisateur dans l'expression de requêtes décisionnelles. Mais compte tenu du volume et de la complexité des données, les utilisateurs ont des difficultés pour exprimer ces requêtes. Ce type de recommandation offre donc deux fonctionnalités principales : d'une part, elle assiste les utilisateurs dans la définition de leurs requêtes en leur proposant des parties de requête pertinentes ; d'autre part, elle propose des requêtes complètes pour faciliter l'exploration des données. Ainsi, cette fonctionnalité vise à améliorer l'efficacité et la précision des requêtes formulées par les utilisateurs, en leur offrant des suggestions pour formuler des requêtes plus complètes et plus ciblées.

De nombreux travaux récents ont étudié l'utilisation des journaux de requêtes. Une proposition intéressante est l'approche QueRIE [73] qui repose

sur une méthode basée sur le filtrage collaboratif ; les sessions de requêtes SQL sont enregistrées et utilisées pour générer des recommandations. Les interactions dans QueRIE sont des séquences de requêtes SQL appelées sessions. Les sessions sont représentées sous forme de matrices répertoriant les requêtes exécutées pendant chaque session. La factorisation de matrices est ensuite appliquée pour découvrir des relations entre les requêtes et les utilisateurs. Lorsqu'une nouvelle session est créée, une approche KNN[14] est utilisée avec des similarités classiques de Jaccard ou de Cosinus pour trouver les requêtes les plus proches de cette session dans la matrice. Ceci permet de recommander des requêtes qui sont susceptibles d'intéresser les utilisateurs en se basant sur les préférences des utilisateurs similaires. Dans [74] les auteurs présentent une méthode de recommandation de requêtes SQL qui vise à faciliter l'exploration des BD relationnelles. Cette méthode assiste les utilisateurs et les aident à formuler des requêtes plus rapidement et plus précisément. Pour y parvenir, YmalDB analyse les requêtes précédemment exécutées ainsi que les résultats obtenus. Ces informations sont ensuite utilisées pour créer un modèle de recommandation capable de suggérer des requêtes similaires qui peuvent être utiles à l'utilisateur. Dans l'article de Khoussainova et al. [75], un système appelé SnipSuggest est proposé pour compléter les requêtes SQL pendant leur saisie, selon le principe de l'autocomplétion [76]. Son objectif est d'améliorer l'efficacité et la convivialité des interactions avec les BD en fournissant des suggestions contextuelles aux utilisateurs pendant la formulation de leurs requêtes. SnipSuggest exploite l'historique des requêtes précédentes pour identifier des corrélations entre les différentes clauses SQL fréquemment utilisées. Ces corrélations sont ensuite utilisées pour générer des extraits qui peuvent être pertinents pour la requête en cours de saisie. Lorsque l'utilisateur commence à saisir une requête, SnipSuggest analyse les clauses partiellement saisies et compare ces fragments de requête avec les corrélations identifiées dans l'historique. Sur la base de ces correspondances, le système propose des extraits de requêtes SQL pertinents qui complètent la requête en cours. L'avantage de cette approche est qu'elle permet aux utilisateurs de formuler rapidement et facilement des requêtes SQL en profitant de l'historique des requêtes précédentes. Les suggestions contextuelles fournies par SnipSuggest aident les utilisateurs à éviter les erreurs de syntaxe et à accélérer le processus d'exploration de la BD. Par ailleurs, l'article [77] décrit différentes techniques pour recommander des requêtes à un utilisateur lorsqu'il saisit des mots-clés dans un moteur de recherche. Les recommandations sont basées sur l'historique de recherche des utilisateurs, l'analyse sémantique des mots-clés et la compréhension de l'intention de recherche de l'utilisateur. Ainsi, en tenant compte des préférences et des comportements de recherche individuels, le système peut fournir des recommandations plus adaptées à chaque utilisateur. En entrée, le système

[14] K-Nearest Neighbor(KNN) Algorithm: https://www.ibm.com/fr-fr/topics/knn

reçoit les données du profil social de l'utilisateur (par exemple, son genre et sa ville d'origine) et la requête postée par l'utilisateur. Sur la base de ces données, une matrice de similarité est générée pour trouver les utilisateurs les plus proches de l'utilisateur actuel. Après cette étape, s'il y a plusieurs utilisateurs concurrents, un classement est effectué. Enfin, une recherche dans le journal des requêtes permet de trouver les requêtes avec les mots clés les plus proches de ceux soumis par les utilisateurs concurrents. De même dans Drushku et al. [78], les auteurs proposent une approche de recommandation basée sur les intérêts des utilisateurs. Des algorithmes de similarité sont utilisés pour identifier les intérêts des usagers en analysant leurs historiques de requêtes. Ceci permet de détecter les intentions de chaque utilisateur et de lui recommander des requêtes.

Des travaux se concentrent sur la recommandation de requêtes OLAP [79], [72], [80]. Cependant, il est important de noter que ces travaux impliquent la modélisation multidimensionnelle de l'entrepôt, une exigence qui diffère de notre propre contexte. En d'autres termes, les recherches précédentes se sont principalement axées sur des scénarios où la modélisation multidimensionnelle de l'entrepôt de données était nécessaire pour la recommandation de requêtes OLAP. Dans notre cas, cette exigence particulière ne s'applique pas, ce qui souligne une distinction significative par rapport aux travaux existants dans le domaine.

Au vu des travaux portant sur l'assistance à l'accès aux données que nous avons présentés, nous constatons que les propositions faites sur les techniques de personnalisation et de recommandation améliorent sensiblement le requêtage par des non informaticiens. Cependant, la spécificité des acteurs-métier concernés par notre cas d'étude médical nécessite le développement de techniques plus simples et plus évolutives que ce type d'utilisateurs peut s'approprier.

2.3. Synthèse de l'état de l'art

Nous venons de présenter un panorama des principaux travaux qui ont été réalisés dans le domaine de l'extraction et l'accès aux données. Nous avons pu constater que les processus d'élaboration d'un entrepôt NoSQL proposés exigent un niveau d'expertise élevé. Ces processus font intervenir des spécialistes (administrateurs de données, concepteurs, développeurs) tout au long des processus. Autrement dit, la complexité des démarches et des outils logiciels actuels ne permet pas aux acteurs-métier d'être autonomes pour concevoir un schéma de données, créer et charger le magasin puis le personnaliser au fil de l'exploitation. Nos travaux ont donc pour objet de proposer une démarche complète et un environnement logiciel pour permettre à des acteurs-métier d'exploiter un LD.

CHAPITRE 3 PROCESSUS D'UNIFICATION DE SOURCES HETEROGENES DE DONNEES

Un LD est un référentiel de stockage centralisé et évolutif qui permet de stocker de grandes quantités de données brutes provenant de diverses sources, sans avoir à les structurer au préalable. Contrairement à un ED, un LD conserve les données dans leur format d'origine, qu'il s'agisse de fichiers plats, de flux de données en continu, de BD structurées ou non structurées, de journaux d'événements, etc. Il offre une solution flexible pour la gestion de données hétérogènes, volumineuses et variées. Dans le cadre des LD, l'ingestion des données fait référence au processus de centralisation et d'unification des données dans un système cible à partir d'une ou plusieurs sources de données. Elle implique le déplacement des données brutes ou transformées depuis leur emplacement d'origine vers le système de stockage final, en l'occurrence un entrepôt. L'ingestion des données est essentielle pour permettre l'accès, l'analyse et la valorisation des données stockées dans un LD. Elle garantit ainsi la disponibilité des informations actualisées et pertinentes pour les processus d'analyse de données et de prise de décision. Cependant, l'ingestion des données à partir d'un LD dans un entrepôt peut entraîner l'introduction de données de mauvaise qualité. Des problèmes tels que des erreurs, des doublons, des incohérences ou des données manquantes peuvent survenir. Il est essentiel de mettre en place des mécanismes pour assurer le transfert des données en garantissant leur qualité et leur fiabilité.

Ce chapitre est consacré à la présentation d'un mécanisme d'ingestion à partir d'un lac de données. Dans le but de faciliter l'accès et l'exploitation des données stockées dans un LD, nous proposons le processus DLtoDW qui permet d'automatiser l'unification de plusieurs types de BD sélectionnées par l'acteur-métier.

Rappelons que nous avons limité notre étude aux BD relationnelles et aux BD NoSQL orientées-documents.

Notre objectif majeur réside dans la résolution de verrous liés à l'alimentation de l'entrepôt NoSQL. Ceci englobe tout d'abord la transformation de modèles à l'aide de l'approche Model-Driven Architecture (MDA), visant à faciliter la transformation des structures de données au sein d'un entrepôt NoSQL. Un deuxième verrou consiste à orchestrer le transfert efficace de sources de données contenues dans le lacs de données vers une base cible, une tâche cruciale pour maintenir la cohérence des données. Parallèlement, nous abordons la problématique de la réduction de la redondance des données sémantiquement similaires. Enfin, notre contribution s'attaque à la conversion des liens entre BD relationnelles et NoSQL, les transformant en références au sein de la base NoSQL cible.

Pour réaliser ce processus d'ingestion, nous avons défini trois modules qui assurent successivement (1) la transformation des données relationnelles en données NoSQL (module *Transfer*), (2) la fusion des classes considérées sémantiquement "similaires" (module *Merging*) et (3) la conversion des liens

relationnels (clés étrangères) et les liens NoSQL en références (module *Convert*). La figure 3.1 montre les trois modules de notre processus.

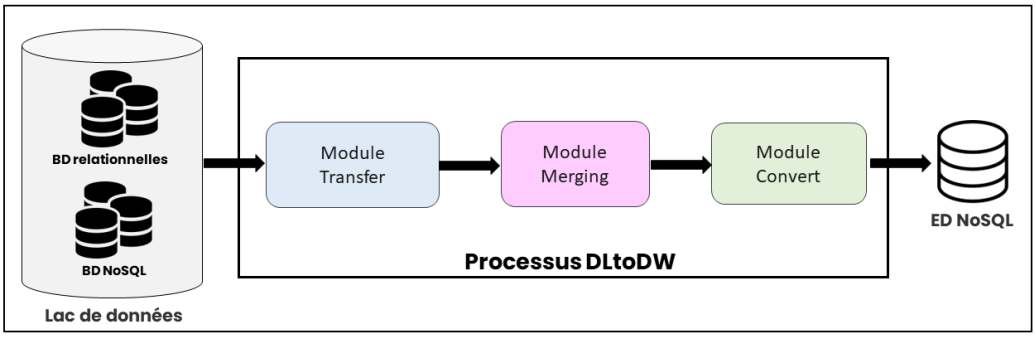


Figure 3.1 – Architecture globale du processus DLtoDW

Dans les sections 3.1, 3.2 et 3.3 de ce chapitre, nous présentons respectivement les trois modules qui constituent le processus DLtoDW. Enfin, dans la section 3.4, nous vérifions ensuite la conformité du transfert des données et nous positionnons notre processus par rapport à l'état de l'art.

3.1. Transfert des données à partir du lac de données

Le module *Transfer* transforme les BD (relationnelles et NoSQL) du LD en une BD unique, appelée entrepôt. Il repose sur l'architecture MDA (Model-Driven Architecture) qui est un standard établi par le consortium OMG[15] pour le développement dirigé par les modèles.

3.1.1. Architecture Dirigée par les Modèles (MDA)

L'Ingénierie Dirigée par les Modèles (IDM ou MDE en anglais) est une approche utilisée par les développeurs pour gérer la complexité des applications informatiques. Cette méthode se base sur la création de modèles et vise à générer une application à partir de ces modèles. Elle constitue un paradigme qui simplifie le processus de développement en fournissant une représentation abstraite de l'application, ce qui facilite la compréhension et la conception de ses différentes composantes.

L'IDM repose sur deux principes fondamentaux. Tout d'abord, elle utilise les modèles comme éléments clés dans le processus de développement. Ces modèles captent les différentes facettes de l'application, telles que son architecture, ses fonctionnalités, etc. Ensuite, l'IDM automatise les transformations entre ces modèles grâce à des règles, permettant ainsi de

[15] The Object Management Group (OMG): http://www.omg.org/

passer d'une représentation abstraite à une représentation plus concrète de l'application. Ces transformations peuvent être réalisées à différents niveaux d'abstraction et permettent de générer automatiquement du code ou d'autres artefacts de l'application.

Pour faciliter l'adoption et l'utilisation de l'Ingénierie Dirigée par les Modèles (IDM ou MDE), diverses spécifications ont été proposées comme l'Architecture Dirigée par les Modèles développée par l'Object Management Group (OMG). Elle repose sur les mêmes concepts que l'IDM, à savoir les modèles et les métamodèles [81], [82]. Les modèles représentent les différentes perspectives d'une application, tandis que les métamodèles décrivent les modèles.

MDA a pour objectif de séparer les spécifications fonctionnelles et les spécifications d'implantation d'une application sur une plateforme donnée [83]. Pour ce faire, elle utilise trois modèles représentant différentes phases du cycle de développement d'une application. Plus précisément, MDA préconise l'élaboration de modèles d'exigences (CIM), d'analyse et de conception (PIM) et de code (PSM). Ces modèles se présentent généralement depuis le niveau le plus abstrait au niveau le plus concret. Comme le montre la figure 3.2, on considère trois grands types de modèles :

- CIM (Computation Independent Model) est le premier modèle dans la chaîne de transformations de MDA. Il décrit les exigences fonctionnelles et non fonctionnelles du système sans tenir compte de la technologie sous-jacente. Le CIM se concentre sur le domaine d'application et ne contient pas de détails sur la mise en œuvre du système.
- PIM (Platform Independent Model) ou modèle indépendant de la plate-forme décrit le système de manière abstraite et indépendante de la plate-forme. Il utilise des concepts de haut niveau pour décrire les fonctionnalités et les comportements du système, sans tenir compte des détails de mise en œuvre. Le PIM peut être transformé en différents modèles de plate-forme, qui sont spécifiques à une technologie particulière.
- PSM (Platform Specific Model) ou modèle spécifique à la plate-forme décrit le système de manière détaillée, en utilisant les technologies spécifiques à la plate-forme choisie. Il contient des détails sur la mise en œuvre du système, tels que le code source, les classes, les interfaces, les bibliothèques et les services spécifiques à la plate-forme.

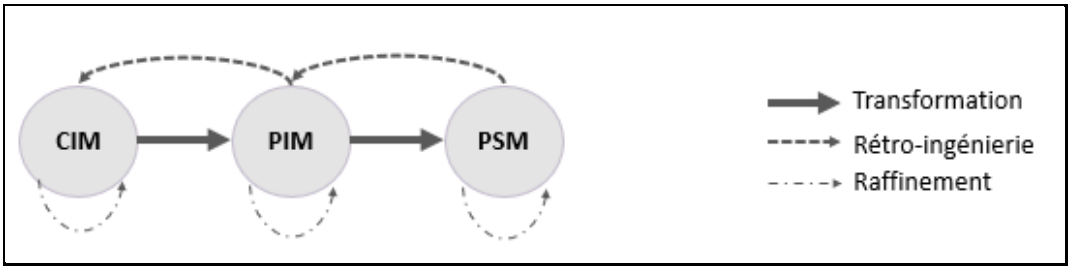


Figure 3.2 – Modèles MDA [84]

MDA utilise la transformation de modèles pour passer d'un modèle à l'autre, ce qui permet une séparation des préoccupations fonctionnelles et d'implantation et une flexibilité accrue dans le développement de logiciels. Ces transformations sont appliquées en utilisant un ensemble de règles qui décrivent comment dériver un modèle cible à partir d'un modèle source. Il existe deux types de transformations de modèles :

- La transformation d'un modèle vers un autre modèle (M2M) : Cette transformation crée un nouveau modèle à partir d'un modèle existant. Par exemple, un modèle PIM peut être transformé en un modèle PSM spécifique à une plateforme donnée.
- La transformation d'un modèle vers du texte (M2T) : Cette transformation génère du code ou du texte à partir d'un modèle. Par exemple, à partir d'un modèle PIM, on peut générer du code source dans un langage de programmation spécifique.

Dans notre cas d'étude, nous sommes amenés à transférer des BD appartenant au LD pour les intégrer dans un entrepôt. MDA permet d'effectuer ce transfert grâce à des règles de transformation. Ces règles doivent être suffisamment générales pour permettre le transfert des données depuis les différentes BD que contient le LD. Elles sont donc spécifiées à partir de métamodèles. Nous avons considéré :

- un métamodèle relationnel pour décrire les BD relationnelles du LD
- un métamodèle NoSQL pour décrire les BD NoSQL stockées dans le LD ainsi que pour décrire l'entrepôt

Ces métamodèles se situent au niveau PIM de l'architecture MDA. Ils décrivent des BD en entrée et en sortie à un niveau logique dans la mesure où les aspects purement physiques (modes d'implantation, table d'index, etc.) ne font pas l'objet d'un transfert.

Pour assurer la transformation d'une BD source vers une BD cible, nous utilisons dans un premier temps, les transformations M2M (Model-To-Model)

pour assurer le passage d'un modèle à un autre puis M2T (Model-To-Text) de MDA pour automatiser le chargement des données dans l'entrepôt. Les règles de transformation sont spécifiées sur les métamodèles décrivant les BD en entrée et la BD en sortie. Pour formaliser ces règles, l'OMG a défini le langage QVT (Query/View/Transformation), qui est devenu un standard de transformation de modèles. Ces règles sont basées sur une syntaxe déclarative qui décrit comment les éléments du modèle source sont transformés pour produire les éléments du modèle cible. Elles permettent de développer des logiciels en accélérant le processus de développement et en facilitant la maintenance grâce à la forme déclarative de QVT. La figure 3.3 montre un exemple de règles QVT qui transforment une BD relationnelle en une BD NoSQL.

Ensuite, nous utilisons une approche par programmation pour assurer la transformation M2T. Celle-ci permet de générer le code nécessaire au transfert des données des BD source vers la BD cible.

```
//Transformer une BD relationnelle en une BD NoSQL
mapping BDR::RDBtoODB(): BDOrientDB{
dbName:=self.DBName;
classes:= self.tables.map toTable();
}

//Transformer une table en une classe
mapping RDB::Tables:: toTable():ODB::Classes{
ClName := self.TableName;
records:= self.rows.map toDoc();
}

//Transformer une ligne en un enregsitrement
mapping RDB::Rows:: toDoc():ODB::Records{
couples:=self.hasAttributes.map toCouple();
}
```

Figure 3.3 – Extrait d'un script QVT pour la transformation d'une BD relationnelle en une BD NoSQL

3.1.2. Caractéristiques du module Transfer

Le module *Transfer* du processus DLtoDW effectue le passage des BD appartenant au LD vers un entrepôt. Ce traitement applique un ensemble de règles de transformation prédéfinies sur une BD. Chaque BD en entrée du module doit être conforme aux concepts spécifiés dans un métamodèle ; nous avons défini deux métamodèles, l'un pour les BD relationnelles et l'autre pour les BD NoSQL.

Le développement du module *Transfer* consiste à définir :

- les métamodèles correspondants à chacun des types de BD contenues dans le LD (BD relationnelles et NoSQL dans notre cas) ; ils permettent

de décrire les modèles propres à chaque BD et ainsi d'extraire les données en ayant identifié leurs caractéristiques

- les règles de transformation des données sources vers les données cibles, ceci pour chaque type de BD.

Ainsi, si l'on veut transférer un nouveau type de BD-sources, il convient de définir le métamodèle correspondant ainsi que de nouvelles règles de transformation ; la BD cible correspondant à l'entrepôt suit toujours un modèle NoSQL.

Soulignons que le transfert d'une BD NoSQL vers un entrepôt NoSQL exige un processus spécifique ; par exemple, dans notre cas d'étude, nous pouvons transférer une BD OrientDB vers un entrepôt OrientDB. Le traitement utilise un même métamodèle mais contient des règles de transformation ad-hoc. En effet, bien que les structures de données entre source et cible ne changent pas, les valeurs des liens (références) de la source doivent être recalculées pour les adapter à l'espace de stockage cible.

Nous considérons deux sous-modules spécifiques dans DLtoDW, *TransferRel* et *TransferNoSQL,* qui sont chargés respectivement de transférer les données d'une BD relationnelle ainsi que les données d'une BD NoSQL comme le montre la figure 3.4.

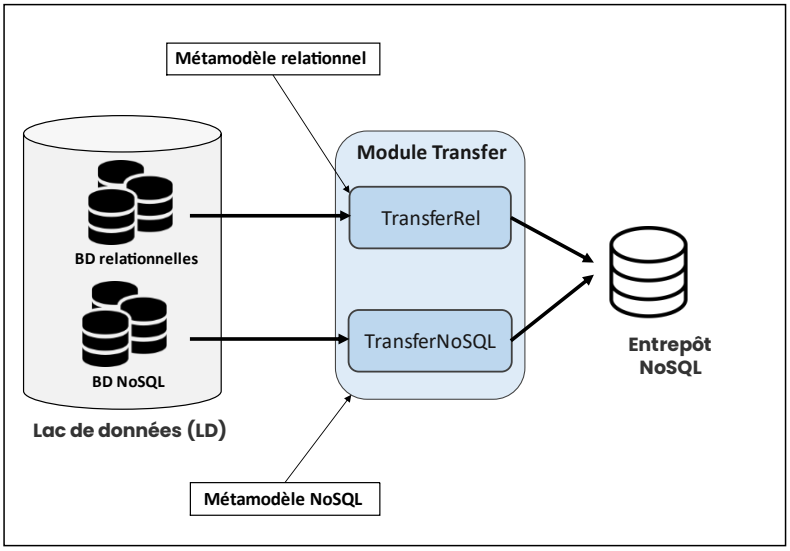


Figure 3.4 – Architecture du module *Transfer*

Il convient de souligner que les modèles relationnels et NoSQL présentent des concepts proches. Le modèle NoSQL est d'ailleurs considéré, pour certains aspects (type des propriétés, liens), comme un sur ensemble du modèle relationnel [85]. Mais notre problématique nécessite de transférer les données contenues dans ces deux types de bases. Or, l'organisation d'une BD

relationnelle est spécifique dans la mesure où schéma et extension sont séparés ; elle n'est donc pas compatible avec la structure d'une BD NoSQL. Ceci ne nous a pas permis d'unifier les deux métamodèles.

Dans les sections suivantes, nous présentons successivement le métamodèle d'une BD relationnelle et celui d'une BD NoSQL. MDA exige que les métamodèles soient décrits en langage XMI[16]. Dans la mesure où XMI s'avère difficile à lire, nous avons utilisé des diagrammes de classes d'UML pour présenter les métamodèles.

3.1.3. Métamodèle relationnel

Une BD relationnelle est composée d'un ensemble de tables. Elle se définit par un schéma et une extension.

Le schéma donne une description des données en termes de tables, d'attributs et de clés.

L'extension correspond aux données elles-mêmes qui sont organisées selon le schéma correspondant ; elle contient des matrices de valeurs atomiques.

Pour décrire formellement les BD, nous utilisons le principe de représentation des objets sous la forme de couples (clé, valeur).

Une BD relationnelle se définit par un couple (Bn, B) où :

- Bn est le nom de la base
- B représente son schéma et son extension : B = (S, E)

Le schéma S définit l'ensemble des tables de la BD. Chaque table est décrite par son nom et un ensemble d'attributs : *S = {(x, A)}*

Dans une table, tout attribut $a \in A$ est un couple (nom, type) où type correspond à un type abstrait de données atomique.

Une table $t \in S$ se définit donc comme suit :

$$t = (x, \{(a_1, d_1), (a_2, d_2), (a_p, d_p)\}) \text{ où:}$$

- a est le nom d'un attribut de la table t
- d le type de l'attribut
- p le nombre d'attributs dans t.

[16] XML Metadata Interchange: https://fr.wikipedia.org/wiki/XML_Metadata_Interchange

La clé primaire d'une table t notée PK^t est constituée d'un groupe d'attributs appartenant à t. Elle se définit comme suit :

$PK^t \in \mathcal{P}(A^t)$ avec :

- $\mathcal{P}$ représentant l'ensemble des parties de A^t
- La fonction Cardinalité(PK^t) $\in$ [1; p].

Une table peut contenir des clés étrangères qui expriment des associations entre les données et qui seront exploitées dans l'étape de transformation de schémas.

$FK^t \subseteq \mathcal{P}(A^t)$ avec $\mathcal{P}$ représentant l'ensemble des parties de A^t.

Toute clé étrangère correspond à une clé primaire dans une table (notée y) qui est déclarée dans le schéma S de la BD.

$$\forall f \in FK^t, \exists y \in S / f = PK^y$$

L'extension E d'une BD correspond aux données contenues dans les tables qui ont été décrites dans le schéma S. Chaque table m dans E contient un ensemble de tuples, notés u, regroupant des valeurs ; chaque valeur correspond à un attribut du schéma de la table.

E = {m} ; m = (x, $\{u_1, u_2, \ldots, u_k\}$) où :

- x est le nom de la table
- u désigne un tuple
- k est le nombre de tuples.

$\forall$ m $\in$ *E, m est en relation avec* t $\in$ S *si et seulement si m et t portent le même nom de table.*

Grâce à cette bijection entre E et S, nous trouvons le schéma de m dans t. De plus, dans la pratique, il existe la même relation d'ordre total sur l'ensemble des attributs dans le schéma d'une table et sur l'ensemble des valeurs de chaque tuple dans l'extension de cette table. Ceci permet d'associer aisément chaque valeur d'un tuple à l'attribut correspondant. Notons que le schéma d'une table s'applique à tous les tuples de cette table. Par conséquent, les nombres d'attributs composant les tuples d'une même table, sont identiques.

La figure 3.5 représente le métamodèle UML d'une BD relationnelle. Elle donne une représentation à la fois du schéma et de l'extension.

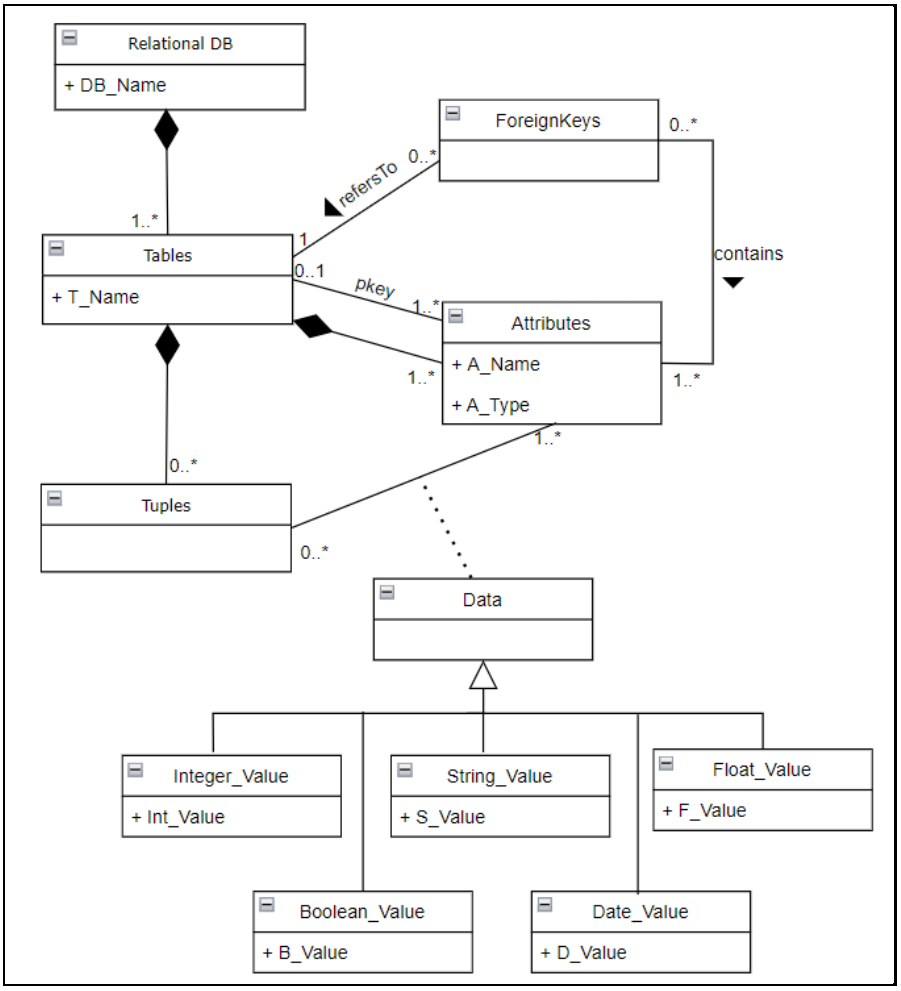


Figure 3.5 – Métamodèle du PIM logique d'une BD relationnelle

3.1.4. Métamodèle NoSQL

Le LD contient aussi des BD NoSQL qui présentent une organisation spécifique liée notamment à la propriété "Schemaless". Le schéma de ces BD est incorporé dans les données et n'est pas directement accessible à l'acteur-métier. Il est donc nécessaire d'extraire ce schéma grâce à des outils qui ont été développés récemment. Par exemple, le logiciel Hackolade[17] permet d'extraire le schéma d'une BD MongoDB. Ainsi, toute BD NoSQL répertoriées dans le LD est associée à son schéma qui a été extrait en amont du processus DLtoDW. Dans ce qui suit, nous formalisons les constituants de toute BD NoSQL, qu'elle soit en entrée ou en sortie de notre processus.

Une BD NoSQL se définit sous la forme d'un couple (Bs, D), avec :

[17] https://hackolade.com/

- Bs le nom de la BD,
- D son contenu.

D contient un ensemble de classes d'objets. Chaque objet (appelé aussi Document ou Enregistrement) se compose de propriétés de la forme (nom, valeur).

D = {C}

Considérons une classe C ϵ D de la forme (x, R) où :

- x est le nom de la classe
- R désigne un ensemble d'enregistrements

$\forall$ r ϵ R, r = (Id, $\{(p_1, v_1), (p_2, v_2), (p_m, v_m)\}$) où :

- Id est l'identifiant (système) de r
- p est le nom d'une propriété
- v sa valeur
- m le nombre de propriétés dans r

Les associations entre les objets sont exprimées sous la forme de propriétés particulières dites de référence. Une propriété de référence ne contient pas, comme dans une BD relationnelle, une valeur d'attribut (clé étrangère) mais un pointeur vers un enregistrement ; plus précisément, une référence est l'identifiant d'un enregistrement dans une classe. Une telle propriété peut être monovaluée ou multivaluée ; dans ce dernier cas, elle contient plusieurs identifiants pour une même classe. Une propriété de référence contient un couple constitué par le nom d'une classe et l'identifiant d'un enregistrement dans la classe.

Considérons un enregistrement $r \in R$ de la forme (Id, P) ; une propriété P = (p, v) est une référence si v = (x, {Id}) où :

- x est le nom d'une classe de la BD
- Id est l'identifiant système d'un enregistrement dans la classe x.

Le métamodèle la figure 3.6 montre que toute classe regroupe des objets (Document ou Enregistrement). Chaque objet a un identifiant et contient un ensemble de propriétés sous la forme de couples (Name, Value). Une valeur est définie par un type atomique, multivalué ou structuré. Dans une BD NoSQL, on distingue deux formes de références selon le système de gestion de la BD source ; le type REF est alors préfixé par « DBRef _ » ou « RID_ ».

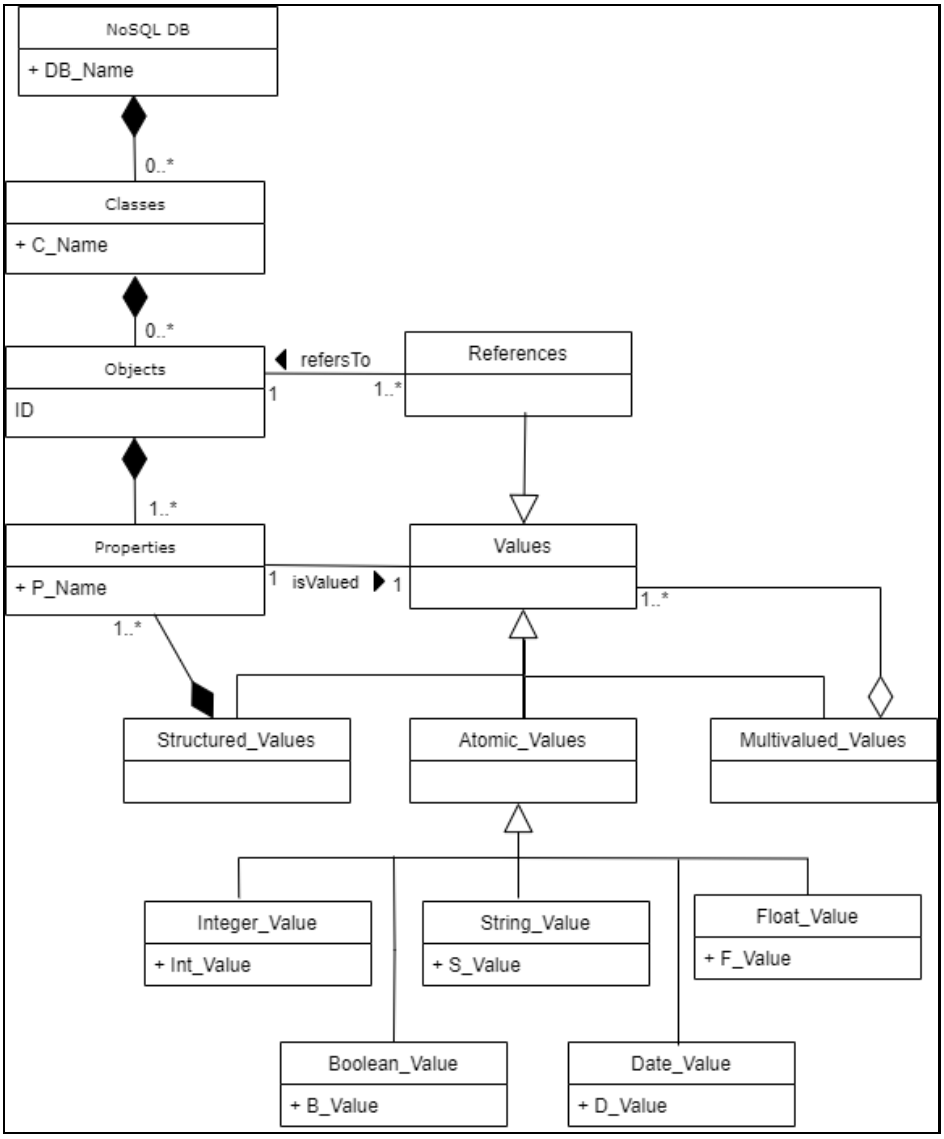


Figure 3.6 – Métamodèle du PIM logique d'une BD NoSQL

3.1.5. Transfert des données relationnelles

Le sous-module *TransferRel* de la figure 3.4 transfère les données de chaque BD relationnelle du LD vers un entrepôt NoSQL unique selon la démarche MDA. Ce transfert est réalisé par l'application d'une suite de règles de transformations sur une BD. Chaque règle a été spécifiée dans le respect du métamodèle d'entrée (relationnel) et du métamodèle de sortie (NoSQL). Pour faciliter leur lecture, nous avons fixé le vocabulaire utilisé en entrée et en sortie conformément aux métamodèles.

BD relationnelle	BD NoSQL
Table	Classe
Tuple	Enregistrement
Colonne	Propriété
Clé primaire	Identifiant
Clé étrangère	Référence

Tableau 3.1 – Correspondance des concepts relationnels et NoSQL

Nous avons vu que dans une BD relationnelle, le schéma est stocké séparément de l'extension (données). Nous disposons donc de deux fichiers XMI, le fichier de la métabase contenant le schéma relationnel d'une part et le fichier de l'extension contenant les données d'autre part. Nous présentons ici informellement les règles qui ont été exprimées en langage QVT :

- **Règle 1 :** Chaque table de la métabase est transformée en une classe. Pour éviter la synonymie, le nom de la classe est préfixé par le nom de la BD.
- **Règle 2 :** Chaque tuple d'une table enregistrée dans l'extension est transformée en un enregistrement dans la classe cible correspondante ; les noms de propriétés sont extraits de la métabase. L'enregistrement contient alors un ensemble de couples (propriété, valeur).
- **Règle 3 :** La clé primaire est stockée dans une propriété dont le nom est étiqueté « PK_ ». A ce stade, les clés étrangères sont également stockées avec leurs valeurs relationnelles associées à une étiquette « FK_ » ; elles seront converties par la suite sous la forme de références par le module *Convert* (§ 3.3).

Ces trois règles s'appliquent pour chaque BD relationnelle du LD. L'entrepôt résultant contient un ensemble de classes. A ce stade, chaque classe contient des enregistrements intégrant leurs schémas. Plus formellement, considérons un enregistrement r dans une classe C :

$\forall$ r $\in$ *C*, r est de la forme (REF, P) avec :

- REF l'identifiant de r unique dans l'entrepôt
- P un ensemble de propriétés de la forme {(Name, Value)} avec *cardinalité(P)* >=2

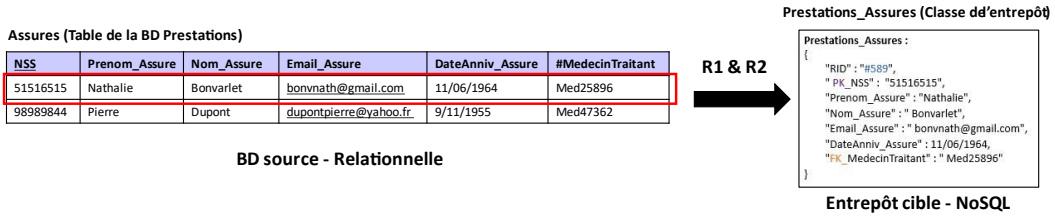
Après application de la règle 2, l'ensemble des propriétés P se compose :

- d'une propriété correspondant à la clé primaire du tuple source correspondant
- le cas échéant, d'une ou plusieurs propriétés contenant les clés étrangères extraites du tuple source.

Ainsi, après l'application des règles pour la transformation de métamodèles, un code est généré basé sur le PIM de l'entrepôt ainsi que le schéma de la BD source pour extraire et charger les données de la source dans cible.

Il convient de rappeler que l'entrepôt n'est pas organisé selon un modèle multidimensionnel et est stocké dans une BD NoSQL orientée-documents gérée par le système OrientDB.

Nous donnons un exemple de transformation d'une table relationnelle en une classe NoSQL.



Dans l'exemple ci-dessus, la colonne NSS de la table « Assures » représente une clé primaire, elle est donc préfixée par « PK_ » dans le nouvel enregistrement. De même, la colonne « MedecinTraitant » est une clé étrangère, elle est précédée par « FK_ » lors du transfert. Toutefois, sa valeur n'est pas convertie dans la nouvelle classe de l'entrepôt. Ceci sera assuré par le module *Convert*.

3.1.6. Transfert des données NoSQL

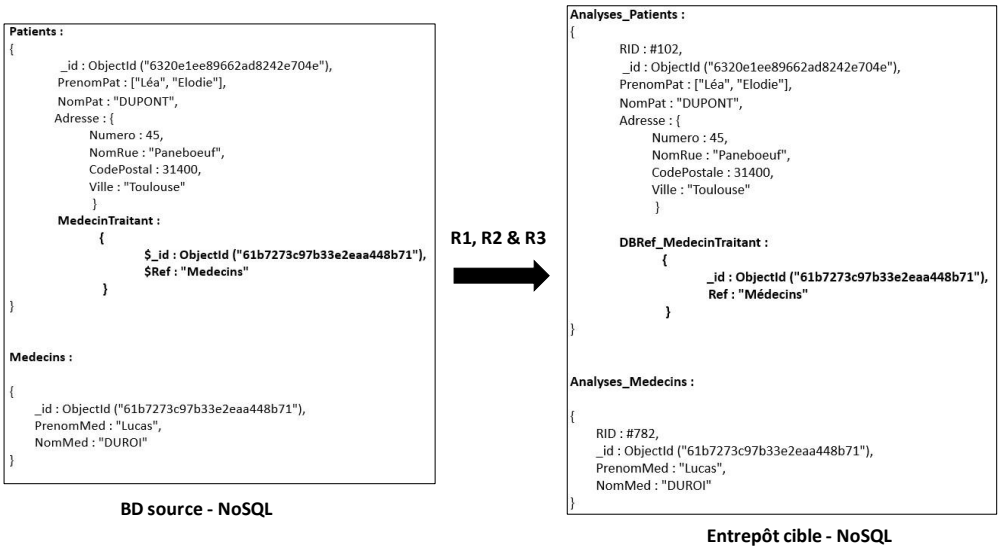
Ce traitement effectue le transfert des BD NoSQL vers l'entrepôt qui est lui-même de type NoSQL orienté-documents. Dans ce type de BD, les liens d'association sont exprimés par des références ; mais, celles-ci s'expriment sous des formes spécifiques selon les systèmes. Les règles que nous proposons ne peuvent donc pas être généralisées mais elles doivent prendre en compte ces spécificités. Dans le cas où la BD source et la BD cible sont gérées par le même système (OrientDB par exemple), le transfert ne nécessite pas de traitements importants à l'exception des liens. En effet, l'espace mémoire de la cible qui sert de référence aux calculs des liens étant différent de celui de la source, il est nécessaire de transformer les liens de la source.

Le transfert des données suit les règles suivantes :

- **Règle 1 :** les noms des classes sont transférés dans l'entrepôt et sont préfixés par le nom de la BD source ; ceci permet d'éviter la synonymie possible des noms de classes dans l'entrepôt.
- **Règle 2 :** chaque enregistrement de la source est transféré dans la cible en préservant la forme initiale. L'identifiant de l'enregistrement source est ajouté dans la BD cible ; il est mémorisé en l'état dans une propriété spécifique dont le nom est « Id ».
- **Règle 3 :** les liens contenus dans les enregistrements sont gérés de manières différentes. Nous distinguons deux cas :
    - cas d'un lien *DBRef* de MongoDB : le nom de la propriété est préfixé par le terme « DBRef_ ».
    - cas d'un lien OrientDB : le nom de la propriété est préfixé par « RID_ ».

Les valeurs des références sont conservées dans leur état d'origine. Dans un second temps, ces valeurs seront transformées en références dans l'espace mémoire de l'entrepôt par le module *Convert*.

Prenons l'exemple de la transformation d'un extrait d'une BD MongoDB en une BD OrientDB. Notre processus applique les règles R1, R2 et R3 sur tous les champs des collections « Patients » et « Medecins ». Celles-ci sont préfixées, dans l'entrepôt, par le nom de la BD source « Analyses ». Les valeurs multivaluées et structurées (propriétés « PrenomPat » et « Adresse » par exemple) sont transférées sans modifications. Le champ « MedecinTraitant » étant de type *DBRef*, il est considéré comme une valeur structurée et transféré en l'état.



3.2. Fusion des classes similaires

Le LD est généralement constitué d'un ensemble de BD distinctes gérées indépendamment. L'entrepôt qui résulte du module *Transfer* (décrit dans la section 3.1), peut contenir des ensembles de données « similaires ». Deux ensembles de données sont dits « similaires » s'il s'agit d'ensembles de données qui ont des significations et des contextes similaires. Cela signifie que même si les données sont collectées à partir de sources différentes, elles sont étroitement liées par leur sens et leur usage. Ainsi, dans notre application médicale, des descriptifs au sujet d'assurés ou des listes concernant des médecins figurent dans plusieurs BD du LD. Ces données peuvent s'appliquer aux mêmes entités du monde réel mais ne présentent pas nécessairement les mêmes structures (propriétés différentes).

L'entrepôt peut donc inclure des sous-ensembles contenant des classes liées par la relation d'équivalence « décrivent les mêmes entités » ; chaque sous-ensemble est dénommé groupe d'équivalence. Par exemple, dans l'entrepôt ENS de notre application, les classes B1.Medecins, B2.Therapeutes et B3.Praticiens constituent un groupe d'équivalence (les préfixes B1, B2 et B3 correspondent aux noms des BD d'origine dans le LD).

La problématique qui consiste à fusionner des BD existantes a fait l'objet d'importants travaux de recherche pour gérer des BD réparties [86], [87]. Dans ce même contexte, les recherches de similarité entre propriétés ont permis d'établir des correspondances entre des données provenant de sources différentes [88]. Cependant, le problème de similarité entre classes et la fusion des données qui en découle, s'avèrent spécifiques à notre cas d'étude. Pour décrire ce problème, notons DW = $\{C_1, C_2, \ldots, C_n\}$ l'ensemble des n classes contenues dans l'entrepôt. DW peut inclure p groupes d'équivalence de la forme G = $\{G_1, G_2, \ldots, G_p\}$; chaque groupe G_i correspond à un sous-ensemble de DW avec i $\in \{1,2,\ldots,p\}$ et $p \leq n/2$. Une même classe de DW ne peut appartenir à deux groupes distincts :

$$\forall\, i, \mathrm{j} \in \{1,2,\ldots,p\}, G_i \cap G_j = \emptyset.$$

Concrètement, si nous considérons deux classes C_k et C_m appartenant à un même groupe d'équivalence, ces classes contiennent des enregistrements (objets) représentant des entités de même sémantique. C'est le cas, par exemple, des classes B1.Médecins et B2.Thérapeutes. Le module *Merging* est une fonction qui transforme chaque groupe d'équivalence du DW, quel que soit le nombre de classes qu'il contient, en une nouvelle classe X. Il s'applique à l'ensemble des groupes déclarés sur le DW.

$$\forall\, i \in \{1,2,\ldots,p\}, Merging(G_i) = X_i$$

Pour chaque groupe d'équivalence du DW, le module provoque la suppression de toutes les classes du groupe et leur substitution par la classe X résultante.

La transformation de G en X repose sur l'utilisation d'une ontologie de domaine alimentée par des experts-métier ; l'ontologie proposée permet d'élaborer automatiquement la classe X. Les experts (administrateurs, gestionnaires, acteurs-métier, etc.) possèdent une connaissance approfondie d'une ou plusieurs BD contenues dans le LD. Il leur est demandé de spécifier les éventuelles correspondances sémantiques entre les données des différentes BD. Par exemple, dans le LD ENS, trois BD contiennent des données décrivant des personnes physiques assurées par des mutuelles. L'ontologie indiquera une relation d'équivalence entre ces données. Les relations de similarité entre des données sont enregistrées dans une ontologie, intitulée « Onto », sous la forme d'un graphe ; elles sont obtenues à partir des spécifications fournies par les experts-métiers. Sont ainsi stockés les groupes d'équivalence, les classes qui les composent ainsi que les correspondances entre les identifiants d'enregistrements et les propriétés.

En utilisant UML, la BD « Onto » est organisée comme le montre le schéma de la Figure 3.7.

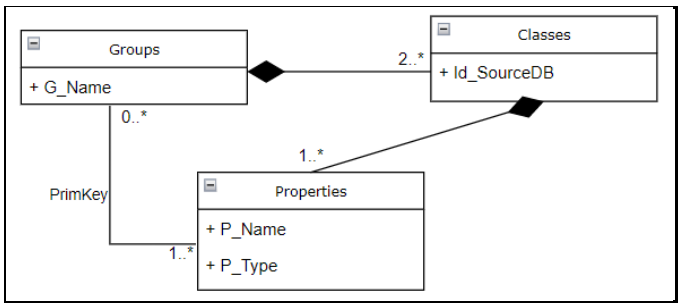


Figure 3.7 – Schéma de la BD « Onto »

Les experts-métiers doivent définir une clé primaire (au sens du modèle relationnel) parmi les attributs des classes d'un même groupe. Cette clé, constituée d'un ensemble minimal de propriétés, permet de distinguer les enregistrements et d'établir les correspondances inter-classes au sein d'un groupe. Par exemple, les classes d'un groupe contenant des descriptifs de médecins ont pour clé le numéro d'identification national professionnel des médecins. Cette clé permet de regrouper les enregistrements concernant une même entité du monde réel. Par ailleurs, les experts examinent les autres propriétés présentes dans les classes et indiquent s'il est opportun ou non de les conserver dans l'entrepôt (classe Propriétés). Ces propriétés sont transférées par le module *Merging* dans la classe X résultante (figure 3.8).

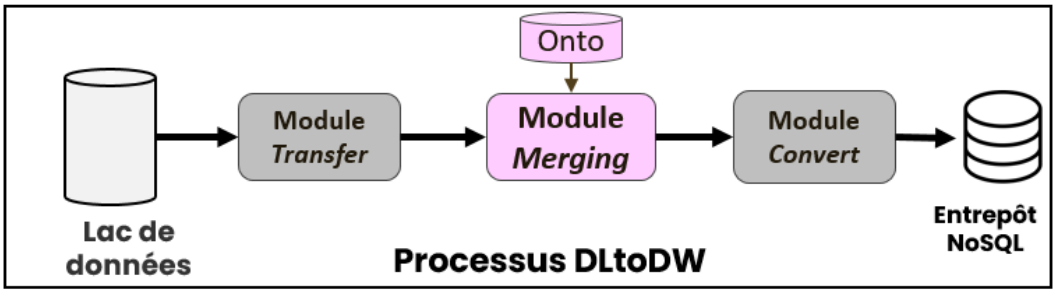


Figure 3.8 – Aperçu du module *Merging*

Pour transformer les classes d'un groupe en une classe unique X, un transfert des valeurs du groupe vers X doit être spécifié. Ainsi, pour chaque propriété à transférer, les experts indiquent dans l'ontologie la classe du groupe qui constitue la source du transfert et qui est susceptible d'assurer une qualité des données optimale ; il s'agit donc de la classe du groupe qui est considérée par les experts comme la plus fiable. Cette source peut varier pour les différentes propriétés de X. Le module *Merging* de notre processus vise à substituer chaque groupe de classes d'équivalence par une classe unique. Nous disposons d'une part de l'ontologie « Onto » et d'autre part de l'entrepôt. L'ontologie décrit la composition de chaque groupe ainsi que les caractéristiques de la classe résultante X. Un extrait de l'ontologie « Onto » est donné dans le chapitre 5. L'entrepôt contient les classes qui seront fusionnées par le module *Merging*.

Algorithm 1 : *Merging*

Input : DW, Onto
Output : DW

```
1   forEach G in Onto.Groups do
2       Create class X in DW            //X will contain the fusion of the classes of G
3       PK ← Onto.PrimKey(G)            //Extracting the primary key of G
4       forEach PK.Value in DW do
5           forEach c in G do           // Similar classes composing G
6               forEach p in c.Properties do    //Property to extract from class c
7                   Extracted ← Extract p.Value from DW
8                   X.AddRecord(PK.Value, p.Name, extractedValue)
9               endFor
10          endFor
11          DW.InsertClass(X)           /Adding a record in X
12      endFor
13      DW.DeleteClass(c)
14  endFor
```

L'algorithme 1 transforme chaque groupe (G) de l'ontologie en une classe unique (X) stockée dans l'entrepôt. La clé du groupe G (commune à toutes les classes similaires) va permettre de rechercher les propriétés à extraire des classes composant le groupe. La nouvelle classe X contient un nombre d'enregistrements égal au nombre de valeurs de la clé dans l'entrepôt. Ce traitement n'est pas limité à une opération d'union entre enregistrements ; en effet, des enregistrements distincts concernant une même entité peuvent disposer de propriétés complémentaires qui seront réunis dans un enregistrement unique.

Le module *Merging* produit une nouvelle version de l'entrepôt NoSQL sur lequel est appliqué le traitement *Convert* chargé de transformer les liens.

3.3. Conversion des liens

Dans le standard des systèmes orientés objet[18], les liens sont matérialisés par des références ; par exemple, ce principe est disponible sous forme native dans le système OrientDB. Il est donc nécessaire de convertir les liens quelles que soient leurs formes (clés étrangères ou liens *DBRef*) en référence dans l'entrepôt.

Pour mettre en œuvre la conversion des liens relationnels et NoSQL, nous avons dû développer deux sous-modules distincts (figure 3.9) ; cependant les principes de conversions des liens sont identiques dans les deux cas. En effet, tout enregistrement r stocké dans l'entrepôt dispose d'un identifiant ; notons E_Id l'ensemble de ces identifiants uniques dans l'entrepôt. Tout enregistrement r dispose aussi d'une propriété étiquetée (c'est-à-dire repérée avec un nom particulier) contenant l'identifiant de l'enregistrement source dont il a été extrait (soit une clé primaire soit un identifiant d'un enregistrement) ; notons E_Rid l'ensemble de ces identifiants-source. Il existe donc une bijection de E_Id dans E_Rid que nous noterons Rb : E_Id → E_Rid.

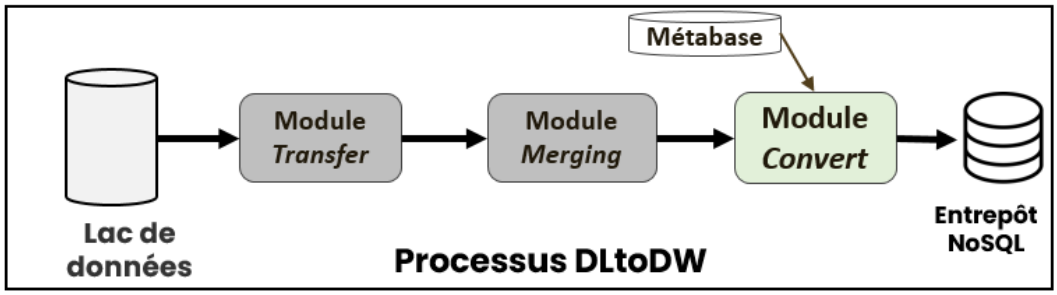


Figure 3.9 – Aperçu du module *Convert*

[18] http://www.odbms.org/wp-content/uploads/2013/11/001.04-Ullman-CS145-ODL-OQL-Fall-2004.pdf

Un enregistrement r peut contenir des propriétés liens qui associent r à d'autres enregistrements. Ces propriétés peuvent avoir pour valeur soit des clés étrangères soit des liens NoSQL. Quoique de formes différentes, ces valeurs sont des identifiants source.

Considérons une propriété-lien dans r de la forme (l, v) où l est le nom du lien, alors sa valeur $v \in E\_Rid$

Si nous notons E_Links l'ensemble des liens contenus dans les enregistrements de l'entrepôt, alors l'application E_Rid → E_Links est injective. Puis, par transitivité, la valeur de l'identifiant système (E_Id) va être substituée à la valeur de l'identifiant-source (E_Links). Dans les deux sections suivantes, nous présentons la mise en œuvre de la conversion des liens relationnels et NoSQL.

3.3.1. Conversion des clés étrangères

Le mécanisme que nous avons développé dans *ConvertRel* repose sur un processus algorithmique. Dans l'entrepôt, tous les enregistrements d'une classe disposent d'un identifiant système (Id).

Lors du transfert des données dans les enregistrements, les clés primaires et les clés étrangères relationnelles ont été reportées en l'état (sans transformation) sous la forme d'une propriété (attribut, valeur), ceci conformément à la règle 2 appliquée par le sous-module *TransferRel* (cf. section 3.1.3). A ce stade, les liens ne sont pas exploitables dans l'entrepôt. L'Algorithme 2 présente le sous-module *ConvertRel* qui convertit les liens relationnels dans l'entrepôt.

Le sous-module *ConvertRel* est donc chargé de substituer chaque clé étrangère en un lien NoSQL dans un enregistrement en parcourant l'ensemble des classes de l'entrepôt.

Algorithm 2 : *ConvertRel*

Input : DW, Metabase
Output : DW

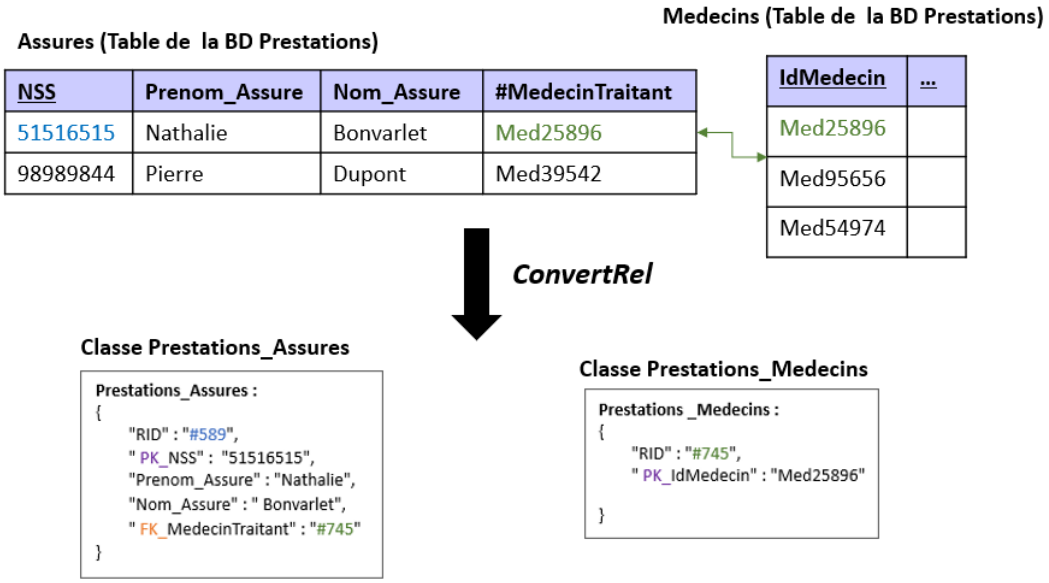
```
1   forEach relation in the metabase do
2     Extract class names Cl1 and Cl2
3     Extract the property names P1 and P2 corresponding to the foreign key
      and primary key (labeled with "FK_" and "PK_")
4      forEach record in Cl1 do
5        If P1.V1 = P2.V2 then
6          Extract Id2 from the record corresponding to V2
7          Assign Id2 to V1
```

8 **endIf**

9 **endFor**

10 **endFor**

Dans l'exemple ci-dessous, l'entrepôt est géré par le système OrientDB. La colonne « MedecinTraitant » représente une clé étrangère étiquetée par « FK_ » dans l'enregistrement. Le système se base sur la métabase relationnelle pour distinguer les clés primaires et étrangères. Le sous-module *ConvertRel* a substitué sa valeur, dans la classe « Assures », avec le RID de l'enregistrement correspondant dans la classe « Medecins ».

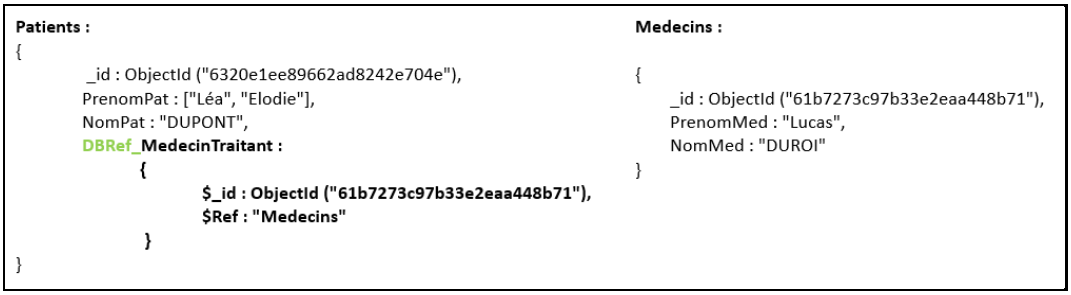


3.3.2. Conversions des liens NoSQL

Dans les BD NoSQL, les liens entre enregistrements peuvent être exprimés au moyen d'un champ structuré, monovalué ou multivalué, repéré par le terme « DBRef_ » pour MongoDB et « RID_ » pour OrientDB. Nous distinguons ainsi deux cas pour la conversion de liens :

Cas de MongoDB

Nous présentons un exemple d'utilisation du champ *DBRef*. Ici, le champ *DBRef* a pour nom « MedecinTraitant » et il établit une relation entre les collections « Patients » et « Medecins ».



Un lien MongoDB contient la collection et l'identifiant d'un document. Il convient de souligner que les liens *DBRef* sont « simulés », c'est-à-dire qu'ils ne sont pas disponibles en mode natif dans MongoDB. Ce système ne gère pas ces champs comme des liens ; l'intégrité référentielle n'est donc pas vérifiée. Lors du transfert des documents dans l'entrepôt, chaque lien MongoDB a été stocké en l'état (sans transformation) sous la forme d'une propriété, ceci conformément à la règle 3 appliquée par le sous-module *TransferNoSQL* (cf. section 3.1.4). A ce stade, les liens ne sont pas exploitables dans l'entrepôt. Le sous-module *ConvertNoSQL* est donc chargé de substituer chaque lien MongoDB par un lien NoSQL.

Nous avons vu qu'un enregistrement r dans une classe C est de la forme (Id, P). P désigne l'ensemble des propriétés de r et chacune des propriétés est de la forme (Name, Value). Une des propriétés de r contient l'identifiant du document MongoDB dont il est issu.

$$\forall\ C \in DW, \forall r \in C, \exists!\ p \in P\ /\ p.Name = Oid\ AND\ p.Value \in\ E\_Oid$$

où E_Oid est l'ensemble des identifiants contenues dans une BD NoSQL du LD.

Avant l'exécution du sous-module *ConvertNoSQL* : parmi les propriétés d'un enregistrement quelconque, on peut trouver un ou plusieurs liens MongoDB. Une telle propriété-lien est étiquetée (c'est-à-dire caractérisée par un nom donnant la sémantique du lien) et contient une valeur structurée précédée du mot-clé « DBRef_ ».

Dans un enregistrement r, une propriété $p \in P$ est un lien ssi

$$p.Name\ contains(\text{“}DBRef\text{”})$$

Le sous-module *ConvertNoSQL* va transformer chaque lien MongoDB en un lien OrientDB grâce à l'application bijective Rb entre E_Oid et E_Rid. Il substituera la valeur Rid ∈ E_Rid au lien MongoDB. L'Algorithme 3 présente le sous-module *ConvertNoSQL* qui convertit les liens dans l'entrepôt. Concrètement, un lien MongoDB est un champ structuré introduit par le mot-clé « DBRef_ » et constitué de trois éléments :

- un couple $id donnant la valeur de l'Oid du document référencé,

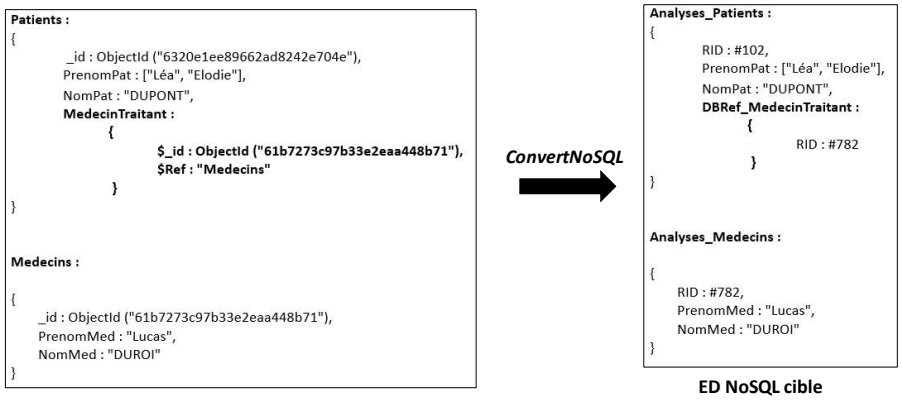
- un couple $ref désignant le nom de la collection contenant le document,
- un couple $db désignant le nom de la BD.

Algorithm 3 : *ConvertNoSQL*

Input : DW
Output : DW

```
1   forEach c in DW do              // scanning of the data warehouse classes
2       forEach r in c do                                   //Each record
3           forEach p in r.P do                             //Each property
4               if p.Name contains "DBRef_" then //if the property is a
                MongoDB link
5                   forEach x in p.Value."$ref" do      //Each record of the
                    referenced class in the DBRef link
6                       if x.Oid = p.Value."$id" then
7                           p.Value ← x.Rid     //substitute the MongoDB link
                            with an OrientDB link
8                       endIf
9                   endFor
10              endIf
11          endFor
12      endFor
13  endFor
```

Ce qui suit montre le résultat de l'exécution de l'algorithme 3 permettant la conversion du champ *DBRef* appelé « MedecinTraitant » dans un entrepôt OrientDB. Ce lien établit une relation entre les classes « Patients » et « Medecins ».



Cas d'OrientDB

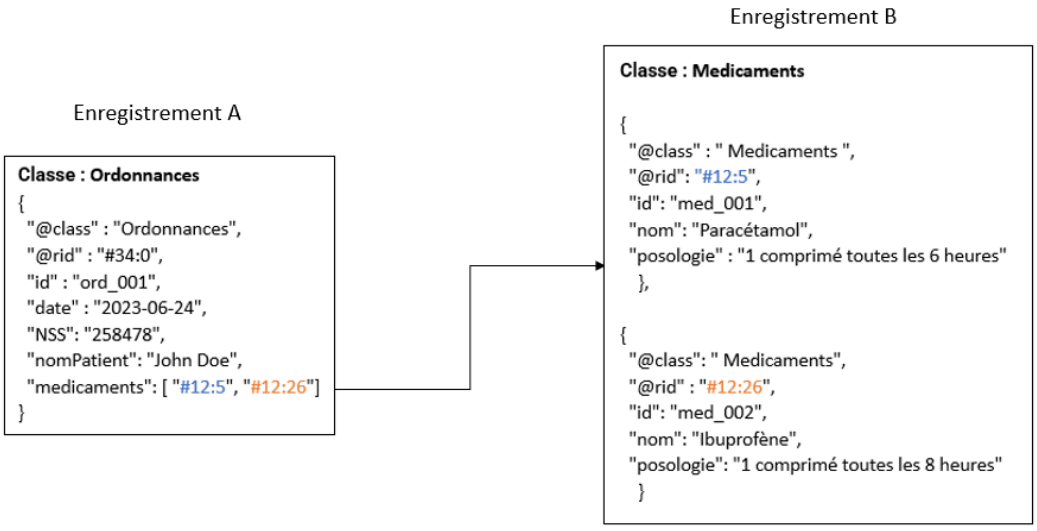
Lorsque de la création d'un enregistrement, OrientDB attribue automatiquement à celui-ci un identifiant unique appelé Record ID (RID) dont la syntaxe est :

#<Identifiant_cluster>:<position> avec :

- Identifiant_cluster donnant le cluster auquel l'enregistrement appartient.
- Position indiquant la position absolue de l'enregistrement dans le cluster.

OrientDB gère les liens en stockant les références des objets cibles de la relation (RID). Prenons l'exemple d'un enregistrement A qui représente une ordonnance. A est lié à l'enregistrement B contenant la liste des médicaments prescrits. Les deux classes « Ordonnances » et « Medicaments » appartiennent à la BD « DossierMedical ». Les deux enregistrements sont liés par la propriété « medicaments » qui permet de les associer dans la BD OrientDB.

La conversion d'une référence OrientDB consiste donc à remplacer la valeur de la propriété (préfixée par « RID_ ») par l'identifiant de l'enregistrement pointé dans la nouvelle classe transférée par le sous-module *TransferNoSQL*.



3.4. Bilan du processus d'ingestion

3.4.1. Synthèse

Dans ce chapitre, nous avons proposé un processus d'ingestion des données d'un LD vers un entrepôt (DLtoDW). Le LD contient plusieurs BD alors que

l'entrepôt correspond à une BD NoSQL unique. Dans notre thèse, nous avons considéré que le LD contient uniquement des BD relationnelles et des BD NoSQL orientées-documents. Trois modules sont mis en œuvre successivement pour assurer l'ingestion des données. Tout d'abord, le module Transfer transforme l'ensemble des BD du LD en une BD NoSQL ; son développement repose sur l'architecture MDA qui offre un cadre formel aux mécanismes de transformation des modèles. Ce transfert des données depuis le LD vers l'entrepôt repose sur des méta-modèles (deux en entrée et un en sortie) et d'un ensemble de règles de transformation exprimées dans le langage déclaratif QVT. Le module Merging fusionne les classes considérées sémantiquement similaires et appartenant à des BD différentes du LD ; cette fusion repose sur une ontologie fournie par des experts-métiers. Enfin, le module Convert traduit les liens relationnels (clés étrangères) et les liens de NoSQL (DBRef de MongoDB et RID d'OrientDB) sous la forme de références conformément aux principes des BD objet. Cette ingestion ainsi que le langage utilisé pour la formaliser le transfert sont présentés dans le chapitre 5 « Expérimentation et Validation ».

3.4.2. Contrôle du transfert

L'ingestion des données dans l'entrepôt suppose que les données ne sont pas dénaturées par la migration effectuée par notre processus. Plus précisément, il convient de vérifier que le transfert des données dans l'entrepôt a été effectué sans perte ni distorsion. Bien entendu, notre processus a modifié les structures de données initiales, a fusionné certains ensembles de données et a transformé les liens. Par conséquent, nous avons mis au point une procédure pour contrôler que les ensembles de données, source et cible, sont équivalents et cohérents au regard des traitements réalisés par les acteurs-métier sur les BD sources (avant ingestion). Nous avons procédé comme suit.

Au sein de la société TRIMANE, en collaboration avec trois informaticiens qui participent au développement de l'application médicale, nous avons constitué un LD test comportant une BD relationnelle et une BD NoSQL avec une intersection de données non vide. Nous avons défini un jeu de trente requêtes portant sur les deux BD sources et représentatif des traitements régulièrement appliqués sur ces BD. Le LD test contenant une BD relationnelle MySQL et une BD MongoDB, Il s'agit donc d'un ensemble de requêtes SQL et NoSQL.

Les requêtes ont été exécutées sur les BD sources et les résultats ont été répertoriés.

Après ingestion de ces deux BD dans l'entrepôt OrientDB, nous avons exprimé l'ensemble des trente requêtes avec le langage de requêtage OrientDB et nous les avons appliquées à l'entrepôt. Les résultats obtenus ont été comparés, requête après requête, avec ceux obtenus sur les BD sources.

Bien entendu, pour une même requête, les structures de données résultantes ne sont pas identiques ; mais les résultats des requêtes sur les sources et sur l'entrepôt ont été jugés équivalents par les développeurs.

3.4.3. Positionnement par rapport à l'état de l'art

A présent, nous situons nos travaux par rapport aux travaux de recherche que nous avons présentés au chapitre 2 et qui ont proposé des mécanismes permettant le transfert de données entre des ensembles de données de types différents.

Les travaux qui ont étudié la conversion d'une BD relationnelle vers une BD NoSQL transforment les tables relationnelles en collections et les lignes en documents. Cependant, les méthodes de conversion de liens diffèrent sensiblement.

Certains travaux cités tels que [53]–[55], [59] utilisent l'imbrication de documents. Cette méthode consiste à stocker les données liées dans un attribut de type multivalué structuré au sein d'un même document. Cette solution est simple à mettre en œuvre mais elle peut entraîner une redondance des données et, par conséquent, des anomalies de mise à jour.

D'autres travaux, notamment [57], [58] privilégient l'utilisation des références (identifiants) plutôt que des imbrications entre les documents. Cette méthode consiste à stocker un identifiant de document dans le document lié. Cela permet de garantir la cohérence des données mais peut être complexe à mettre en œuvre avec le système MongoDB qui ne contrôle pas l'intégrité référentielle.

D'autres travaux combinent les deux méthodes de conversion de liens (par imbrication et par référence). L'article [51] se base sur les cardinalités pour choisir la méthode de conversion adaptée dans la BD NoSQL : l'imbrication de documents (dans le cas de liens un à un et un à plusieurs) ou l'utilisation des références (dans le cas de liens many-to-many). Les auteurs de [60] ont considéré la transformation des liens d'association uniquement.

Certains travaux [61], [62] proposent des règles de transformation pour migrer d'une BD relationnelle vers une BD NoSQL en utilisant l'architecture MDA pour décrire la transformation automatique des modèles.

Concernant le transfert d'une BD NoSQL vers une autre BD NoSQL, on trouve peu de travaux dans la littérature [63], [64]. Dans le cas où les modèles de BD NoSQL sont identiques entre source et cible, le transfert semble évident et on pourrait penser qu'il se limite à un "copier/coller" de la BD source. Dans notre contexte, il n'en est rien puisque la cible du transfert correspond à une nouvelle BD centralisée (l'entrepôt) qui va regrouper l'ensemble des données

extraites du LD. Par conséquent, le transfert nécessite de recalculer tous les liens entre objets dans la mesure où les référentiels des mémoires de stockage source et cible sont différents.

L'état de l'art montre que les solutions actuelles ne répondent que partiellement à notre problématique. D'une part, les solutions proposées ne prennent pas en compte certaines relations entre les données (liens d'association, de composition et d'héritage). D'autre part, ces solutions ne considèrent qu'une seule BD en entrée ; or, un LD contient plusieurs ensembles de données.

Le processus DLtoDW que nous avons proposé répond bien aux exigences de notre application pour transférer les données extraites d'un LD dans un entrepôt supporté par un système NoSQL. Le transfert est effectué selon les principes de l'architecture MDA qui d'une part assure la cohérence du transfert et d'autre part garantit l'évolutivité de l'entrepôt face aux évolutions techniques.

Nous présentons dans le tableau 3.2 une synthèse pour situer nos propositions par rapport aux solutions existantes en utilisant les critères suivants :

- le type des BD source et cible dans le transfert des données,
- l'utilisation de MDA,
- le respect des contraintes d'intégrité,
- l'expression et la conversion des liens dans la cible,
- le traitement des données similaires provenant de sources diverses

Critères / Articles	Transfert des données		Multiplicité des sources	Utilisation de MDA	Conversion des liens		Respect des contraintes d'intégrité référentielle	Fusion des données "similaires"
	Relationnel vers NoSQL	NoSQL vers NoSQL			Par imbrication	Par référence		
[53]	✓	X	X	X	✓	X	X	X
[54]	✓	X	X	X	✓	X	X	X
[55]	✓	X	X	X	✓	X	X	X
[59]	✓	X	X	X	✓	X	X	X
[57]	✓	X	X	X	X	✓	X	X
[58]	✓	X	X	X	X	✓	X	X
[60]	✓	X	X	✓	✓	✓	X	X
[61]	✓	X	X	✓	✓	✓	X	X
[62]	✓	X	X	X	✓	X	X	X
[63]	X	✓	X	X	✓	✓	X	X
[64]	X	✓	✓	X	✓	X	X	X
DLtoDW	✓	✓	✓	✓	✓	✓	✓	✓

Tableau 3.2 – Positionnement de notre processus par rapport à l'état de l'art

CHAPITRE 4 PROCESSUS D'ASSISTANCE

AU REQUETAGE

La Business Intelligence (BI) en libre-service est une approche qui vise à permettre aux utilisateurs métier d'accéder aux données de manière autonome, sans dépendre étroitement des équipes informatiques ou des analystes de données. L'objectif principal de la BI self-service est de donner aux utilisateurs non informaticiens la capacité de prendre des décisions éclairées en exploitant les données de l'entreprise de manière flexible et intuitive.

Dans le contexte de l'évolution vers la BI en libre-service et la nécessité de faciliter l'accès aux données pour les acteurs-métier, la personnalisation et la recommandation émergent comme des leviers stratégiques. Ces concepts offrent une solution pour répondre aux besoins spécifiques des utilisateurs métier en matière d'analyse de données.

Dans la section 4.1, nous abordons la problématique liée à l'accès aux données complexes dans le contexte d'un LD et présentons notre solution. Dans la section 4.2, nous étudions la personnalisation des schémas de magasins pour répondre aux besoins spécifiques d'un acteur-métier. La section 4.3 décrit le processus de recommandation de schémas que nous proposons. Enfin, dans la section 4.4, nous évaluons l'apport de notre solution par rapport aux approches existantes.

4.1. De la problématique à une proposition de solution

L'accès aux données est un élément fondamental dans le domaine de l'informatique décisionnelle. Les organisations collectent et stockent une quantité croissante de données provenant de différentes sources. Cependant, la simple collecte et le stockage de ces données ne suffisent pas à en tirer de la valeur. Pour exploiter pleinement le potentiel des bases et LD, il est essentiel de fournir aux acteur-métiers un accès efficace, pertinent et personnalisé à l'information qu'ils contiennent. L'accès aux données revêt une importance particulière pour les acteurs-métier parce que ceux-ci doivent prendre des décisions éclairées et stratégiques basées sur une analyse approfondie des données disponibles. Cependant, il peut s'avérer difficile à mettre en œuvre en raison de divers facteurs tels que le volume important de données, leur complexité, leur hétérogénéité et leur répartition dans des systèmes de stockage divers. Le manque technique parmi les acteurs-métier constitue une autre facette du défi d'accès aux données. Souvent spécialisés dans leurs domaines fonctionnels respectifs, ces professionnels peuvent être confrontés à des barrières techniques lorsqu'il s'agit d'interagir avec des systèmes complexes de gestion de données. La compréhension des structures de BD, des requêtes SQL, ou d'autres compétences techniques spécifiques peut être limitée, ce qui entrave leur capacité à naviguer efficacement à travers les différents silos de données. Dans ce contexte, les efforts des éditeurs de logiciels et des chercheurs se

sont concentrés sur le développement de techniques et de méthodologies visant à faciliter l'accès aux données. Ces approches cherchent à optimiser la recherche, la récupération et la présentation des données de manière à ce qu'elles soient pertinentes et adaptées aux besoins spécifiques des acteur-métiers.

Dans la section 4.1.1, nous présentons la problématique de l'accès aux données complexes. Comme mentionné précédemment, les acteur-métiers peuvent éprouver des difficultés à formuler des requêtes pour extraire les données. Cette difficulté est souvent liée à la complexité des structures de données et des relations dans l'entrepôt. La section 4.1.2 décrit notre approche.

4.1.1. Accès aux données complexes

Dans notre cas d'étude, le LD ENS regroupe plusieurs bases de données utilisées à des fins décisionnelles. Un entrepôt NoSQL, dédié à la prise de décision, a été créé à partir du LD via le processus DLtoDW (cf. chapitre 3). Si l'on considère la requête suivante posée par un acteur-métier sur l'entrepôt : *"Consulter la liste des consultations de médecins généralistes liées au COVID en France au dernier trimestre 2022"*. Pour répondre à cette requête dans un langage de type SQL, l'acteur-métier doit sélectionner les données pertinentes relatives aux consultations médicales, aux médecins généralistes, au COVID et à la localisation géographique. De plus, il est nécessaire de spécifier la période, à savoir le dernier trimestre de l'année 2022, afin d'obtenir la liste des consultations. Ainsi, l'acteur-métier doit être en mesure d'identifier les sources de données appropriées au sein de l'entrepôt. Cela peut inclure les classes contenant les données des consultations médicales, des médecins, des patients et des informations spécifiques liées au COVID.

Ainsi, compte tenu de la complexité des structures et des liens dans l'entrepôt, on constate que la plupart des acteur-métiers appréhendent mal les données à sélectionner. L'un des verrous à lever concernait donc la difficulté des utilisateurs généralement des acteur-métiers à exprimer leurs besoins d'une manière appropriée. De plus, lors de l'exploitation actuelle de l'ENS, nous avons pu observer une proximité des besoins en données pour les acteur-métiers ayant des activités voisines. Par exemple, les analystes financiers et les contrôleurs de gestion manipulent fréquemment les données des centres de soins, les assurés et les documents transmis par ces derniers. Ce constat nous a conduit à considérer, comme un second verrou, la mise en commun de ressources pour des acteur-métiers partageant les mêmes centres d'intérêts ; ainsi l'acteur-métier n'aura accès qu'aux données qui présentent un intérêt immédiat pour lui. Notre objectif consiste à résoudre les défis liés à la conception de magasins NoSQL adaptés aux besoins spécifiques des acteurs-métiers. Deux concepts majeurs émergent dans ce domaine : la recommandation et la personnalisation de l'accès aux données.

L'enjeu réside dans la proposition de schémas de données en se basant sur la compréhension des activités antérieures des utilisateurs sur les données, visant ainsi à fournir une assistance efficace dans le processus d'élaboration de magasins NoSQL adaptés à leurs besoins spécifiques. Notons à ce stade, que les acteurs-métiers, par définition, maitrisent les concepts contenus dans les données qu'ils manipulent. Par exemple, un professionnel d'une mutuelle appréhende facilement des entités du monde réel telles que des médecins, des assurés, des consultations. Ces entités se retrouvent dans les schémas des BD, de l'entrepôt et des magasins.

4.1.2. Survol de notre solution

Pour assister les acteur-métiers dans l'expression de leurs besoins en données, nous avons développé le processus DWtoDM situé en aval du processus DLtoDW, comme illustré dans la figure 4.1. Ce processus permet aux acteurs-métier de sélectionner les données dont ils ont besoin dans l'entrepôt et de les transférer dans un magasin dédié. Ainsi, chaque acteur-métier ou classe d'acteur-métiers peut ainsi constituer ses propres magasins. Lors de chaque alimentation du LD (en moyenne une fois par semaine), un entrepôt est créé automatiquement à partir de la dernière version du LD ; le schéma de cet entrepôt est appelé schéma global. Au début d'une session, l'acteur-métier peut dériver ou créer ses propres schémas de magasin.

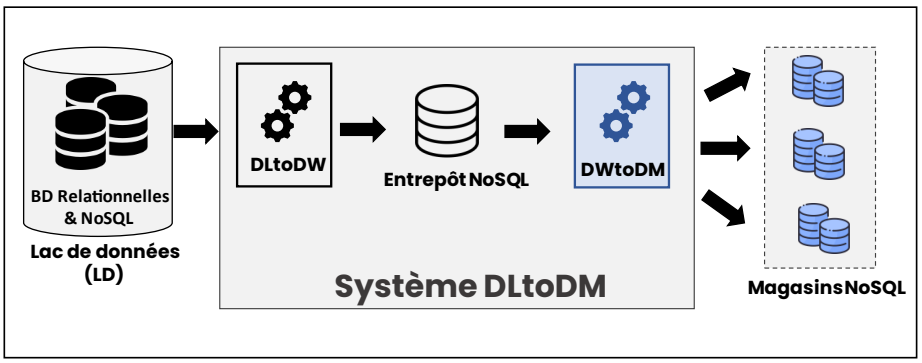


Figure 4.1 – Place du processus DWtoDM

Le processus offre donc à l'acteur-métier trois démarches d'assistance pour obtenir un magasin conforme à ses besoins :

1- le processus propose le schéma global ainsi que les schémas des magasin créés par l'acteur-métier antérieurement ; grâce à des métadonnées sur chaque magasin, l'acteur-métier choisit le schéma qui lui semble le plus proche.

2- l'acteur-métier choisit une classe de données correspondant à son centre d'intérêt pour créer un nouveau schéma. A partir de cette classe notée c, il a la possibilité d'augmenter son schéma du magasin en ajoutant des classes adjacentes à c, c'est-à-dire liées directement à c par des liens d'associations, de composition et d'héritage. Il peut ensuite réitérer cette opération sur d'autres classes du magasin. Cette démarche de personnalisation, appelée Augmentation d'un magasin, est présentée en section 4.2.

3- il exprime ses besoins en données (requête) sous la forme d'une phrase ou d'une suite de mots-clés. Le processus lui recommande alors les schémas des autres usagers jugés les plus proches de ses besoins. Cette démarche est basée sur un principe de recommandation décrit dans la section 4.3.

Les deux premières étapes relèvent d'un mécanisme de personnalisation ; la troisième s'inscrit dans un processus de recommandation.

Pour qu'un acteur-métier puisse gérer efficacement ses schémas, le processus DWtoDM demande que tout schéma soit associé à un nom (mnémonique), un commentaire et des mots-clés.

Le processus DWtoDM propose deux mécanismes complémentaires pour faciliter la création des schémas de magasins. Tout d'abord, un mécanisme de personnalisation permet à l'acteur-métier de créer des schémas de magasin personnalisés à partir du schéma global du DW. Ensuite, un mécanisme de recommandation propose les schémas de magasin existants les plus pertinents en fonction des besoins des acteur-métiers. La figure 4.2 montre les deux mécanismes composants le processus DWtoDM.

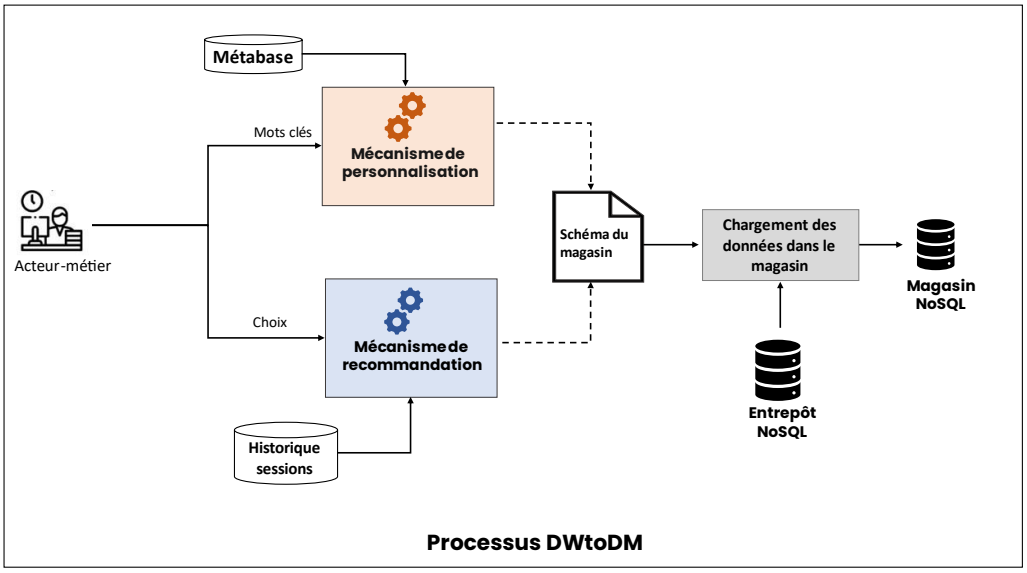


Figure 4.2 – Architecture globale du processus DWtoDM

Une fois le schéma élaboré, le processus est capable de charger les données dans le magasin.

Il est à noter que, dans cette étude, nous n'abordons pas les problèmes liés à la réorganisation des magasins selon un modèle multidimensionnel.

Nos principales contributions sont les suivantes :

1. le mécanisme de recommandation de schémas est basé sur un filtrage collaboratif passif, c'est-à-dire sans évaluations fournies par les acteur-métiers. Il identifie les schémas de magasin pertinents en se basant sur des corrélations entre les acteur-métiers et les schémas existants dans l'entrepôt.

2. le mécanisme de personnalisation offre à l'acteur-métier la possibilité de choisir les classes et/ou attributs de l'entrepôt, permettant ainsi de sélectionner les données appropriées dans l'entrepôt pour créer un ou plusieurs magasin selon ses besoins.

3. les métadonnées décrivant les historiques des sessions sont mises-à-jour à chaque rafraichissement de l'entrepôt. Ceci garantit que les informations sur les acteur-métiers et les schémas de magasin restent à jour, améliorant ainsi la précision des recommandations et la pertinence du processus de personnalisation.

Dans la figure 4.3 ci-dessous, nous présentons le diagramme d'activité UML du processus DWtoDM. Ce diagramme illustre les différentes étapes des processus de personnalisation et de recommandation de schémas de magasins.

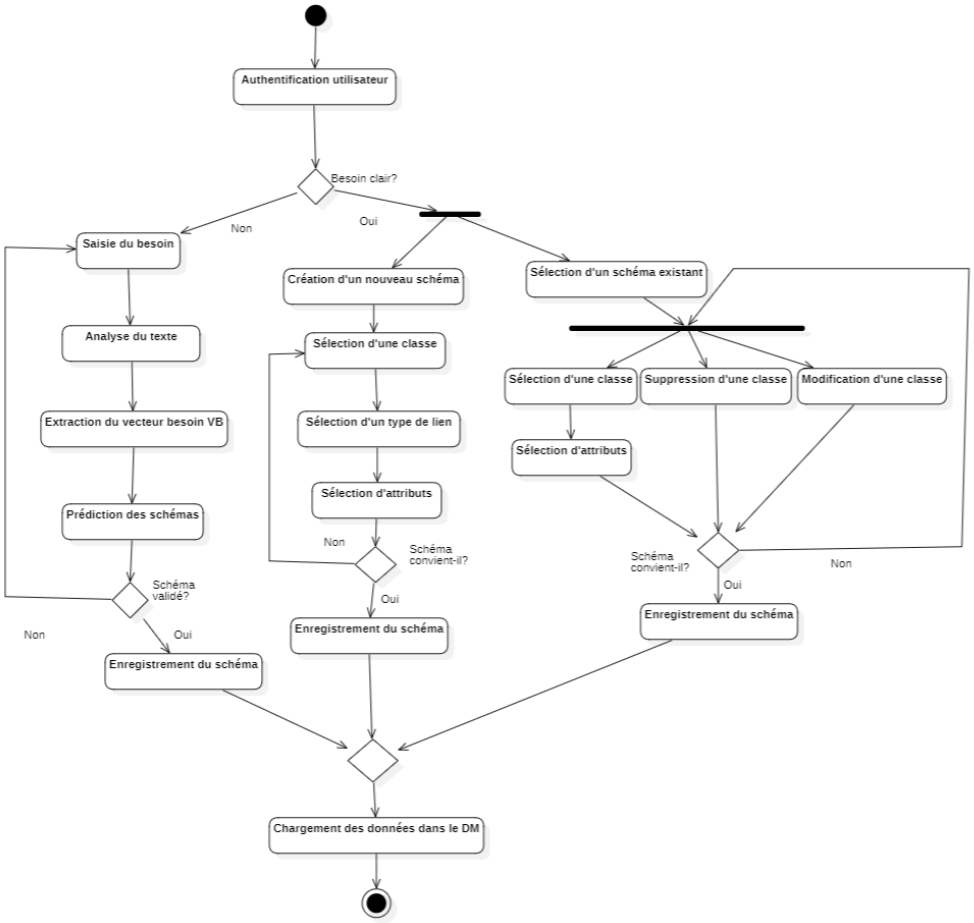


Figure 4.3 – Diagramme d'activité du processus DWtoDM

Dans la section 4.2, nous décrivons le mécanisme de personnalisation. Ce mécanisme permet d'adapter les magasins aux besoins spécifiques des acteur-métiers en prenant en compte leurs préférences et leurs activités antérieures. Il vise à fournir une expérience acteur-métier personnalisée et à faciliter l'exploration des données. Dans la section 4.3, nous mettons en œuvre notre propre mécanisme de recommandation spécifiquement conçu pour faciliter la création de magasins. Ce mécanisme utilise des algorithmes pour analyser les schémas de magasin existants et les interactions utilisateur-système. Il génère ensuite des suggestions de schémas de magasin personnalisés et adaptés aux besoins des acteur-métiers.

4.2. Personnalisation des schémas

La personnalisation de schéma consiste à adapter la structure des données en fonction des besoins et des préférences spécifiques d'un acteur-métier. Ceci permet d'optimiser la pertinence et l'efficacité des données sélectionnées. Notre mécanisme de personnalisation vise à assister l'acteur-

métier dans l'élaboration d'un schéma à partir des informations décrivant son comportement ou ses actions passées.

L'architecture du mécanisme de personnalisation est représentée dans la figure 4.4 :

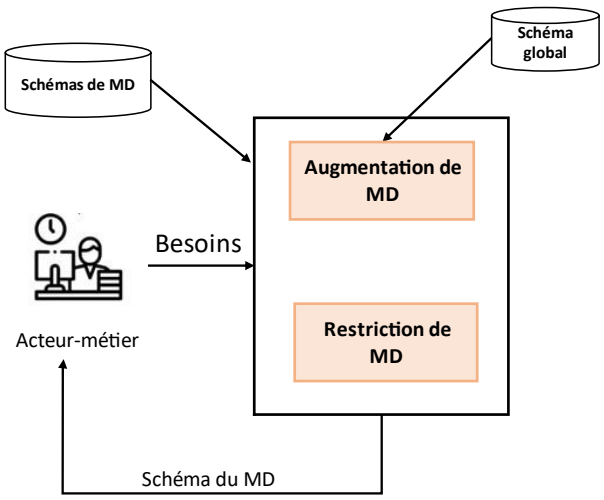


Figure 4.4 – Architecture du mécanisme de personnalisation

4.2.1. Augmentation du schéma d'un magasin

Lors de l'alimentation de l'entrepôt (généralement chaque semaine), une métabase décrivant son schéma est automatiquement créée. Cette métabase contient la description de l'entrepôt et est associée à son schéma appelé schéma global. En raison de sa complexité, le schéma global n'est généralement pas utilisé directement pour exprimer des requêtes mais il permet de générer des magasins correspondants aux besoins des acteur-métiers.

Pour élaborer un nouveau schéma, l'acteur-métier s'appuie sur le schéma global ou le schéma d'un magasin existant. Il sélectionne une ou plusieurs classes correspondant à son centre d'intérêt. Un schéma est un graphe non orienté. Il est défini par le couple P = (N, L) avec N représentant un ensemble de nœuds et L désignant l'ensemble des liens étiquetés qui relient des nœuds de N (liens d'appartenance, d'association, de composition et d'héritage).

N correspond aux noms des éléments constitutifs du schéma de l'entrepôt :

$$N = \{classe\} \cup \{attributs\}$$

Un lien est défini par son étiquette et ses extrémités :

$$L = \{(e, (x, y)) \mid (x, y) \in N \times N\}$$

où l'étiquette e permet de distinguer le type du lien (Appartenance, Association, Composition, ou Héritage). Cette notation permet d'exploiter les matrices d'adjacence qui vont assister les acteur-métiers pour construire progressivement le schéma d'un magasin.

Les matrices d'adjacence sont associées au graphe représentant le schéma global ; elles sont créées immédiatement après la création de l'entrepôt. Elles contiennent donc les nœuds correspondant à la totalité des données enregistrées dans le LD. On dispose d'une matrice symétrique par type de liens (matrices d'association, de composition et d'héritage). Chaque terme $a_{i,j}$ vaut 1 si 2 nœuds sont liés et 0 sinon. La figure 4.5 montre un extrait de la matrice des liens d'association. Si l'acteur-métier sélectionne la classe Assurés, il obtient ainsi toutes les classes liées directement à Assurés par des liens d'association.

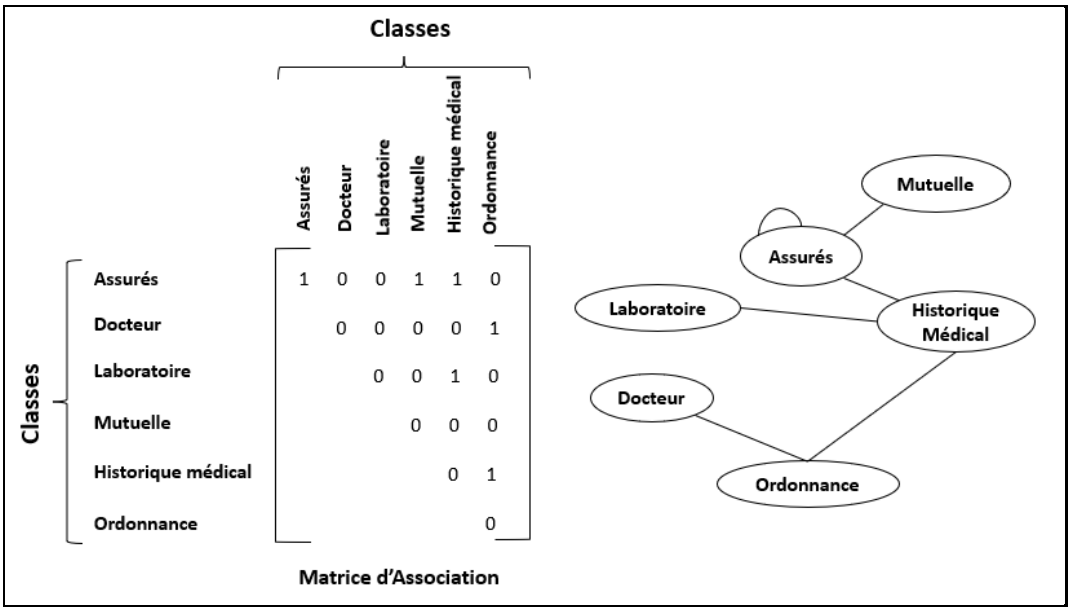


Figure 4.5 – Extrait d'une matrice des liens d'association

Ce mécanisme de personnalisation permet de lever l'un des verrous de notre problématique en offrant la possibilité d'augmenter de manière incrémentale un schéma initial. On peut souligner que le choix des termes désignant les classes, les attributs et les liens est primordial pour exprimer la sémantique des données et faciliter les choix des acteur-métiers. Notons aussi que notre mécanisme ne permet pas de restreindre les objets que contiennent les classes d'un magasin ; seule une limitation du schéma a été considérée utile. Une fois le schéma élaboré par l'acteur-métier, le nouveau magasin est alimenté à partir du magasin qui a servi de base à l'élaboration du schéma (entrepôt global ou autre magasin).

4.2.2. Restriction du schéma d'un magasin

Une autre fonctionnalité de personnalisation est assurée par le mécanisme et proposée à l'acteur-métier à la fin de chaque session. Il s'agit de restreindre le magasin courant aux seules données qui ont été manipulées au cours de la session et d'enregistrer le schéma correspondant. Cette fonctionnalité est justifiée par le constat qu'une session peut durer plusieurs heures ; et les besoins s'affinent généralement tout au long de la session, requête après requête. L'acteur-métier peut vouloir mémoriser un schéma plus précis que le schéma initial, en supprimant les classes non utilisées dans les requêtes. Voici comment ce traitement de restriction est mis en œuvre.

Comme le montre la figure 4.2, les requêtes sont appliquées sur un magasin existant ou généré par l'acteur-métier en début de session. Le module de requêtage (hors cadre d'étude de cette thèse) enregistre les requêtes de l'acteur-métier dans un fichier Log (QLog) qu'il transmet au processus DWtoDM en fin de session. Toute requête d'accès aux données appliquée sur un magasin NoSQL est formulée avec un langage de type SQL[19] (langage de OrientDB pour notre cas d'étude) ; elle est donc de la forme SELECT <attributs> FROM <classes> WHERE <prédicats>. Le processus DWtoDM balaie le fichier QLog de l'acteur-métier et en extrait les noms des classes manipulées par chaque requête. L'opération de restriction consiste à supprimer toutes les classes qui n'ont pas été manipulées dans le schéma courant. Pour les classes restantes, les attributs et les liens du schéma initial (en début de session) sont conservés. Le nouveau schéma peut remplacer le schéma initial ou être enregistré séparément du schéma initial.

Plus précisément, pour chaque session effectuée par un acteur-métier, le fichier QLog contient l'ensemble des requêtes qui ont été appliquées sur le magasin. Toute requête R est définie par un triplet (A, C, P) avec A l'ensemble des attributs retournés par la requête, C l'ensemble des classes manipulées et P l'ensemble des prédicats et leurs connecteurs logiques. Considérons un acteur-métier x ayant réalisé une session avec un magasin de données noté MD^x. Le fichier QLog transmis par le processus de requêtage à DWtoDM se présente sous la forme suivante :

$$QLog^x = \{R_1, R_2, \ldots, R_n\}$$

où *n* représente le nombre de requêtes émises par l'acteur-métier x pendant une session. Les fonctions *RClass* et *MClass* retournent respectivement l'ensemble des classes utilisées par une requête et l'ensemble des classes d'un magasin. Ainsi, toute requête dans le fichier QLog utilise les noms des classes figurant dans le magasin de l'acteur-métier.

[19] https://orientdb.com/docs/last/gettingstarted/Tutorial-SQL.html

$$\forall R_i \in QLog^x \text{ , } RClass(R_i) = C \text{ avec } C \subseteq MClass(MD^x)$$

A la fin d'une session, le fichier QLog est fermé. Le système propose alors à l'acteur-métier de créer une nouvelle version du magasin par restriction des classes.

$$V2\_MD^x = MD^x \cap (\cup_{i=1..n} RClass(R_i))$$

Les deux mécanismes de personnalisation présentés permettent à un acteur-métier (ou une classe d'acteur-métiers) de créer de nouveaux schémas adaptés à ses besoins. Mais ces procédures fonctionnent uniquement sur les schémas des magasin propres à chaque acteur-métier. Cependant, même avec la personnalisation des schémas de magasins, il peut être difficile pour les acteurs-métier de déterminer exactement quelles classes et attributs inclure ou exclure pour répondre à leurs besoins d'accès aux données. C'est à ce niveau qu'un mécanisme de recommandation entre en jeu s'avère utile.

Le mécanisme de recommandation peut utiliser les schémas de magasin comme base de connaissances afin de proposer des suggestions pertinentes. En analysant les schémas existants et en identifiant les similarités entre les profils d'utilisateurs, le mécanisme recommande des classes, attributs ou relations qui pourraient être utiles aux acteur-métiers en fonction de leurs besoins spécifiques.

Dans la section suivante, nous proposons une approche de recommandation de schémas de magasins afin d'assister les acteurs-métier dans la sélection des classes et attributs pour leurs besoins.

4.3. Recommandation de schémas

Le processus de recommandation que nous proposons permet à un acteur-métier de bénéficier des magasins d'autres acteur-métiers. Seront recommandés à un acteur-métier les schémas de magasin qui sont proches de ses besoins en données et ceci quel que soit les acteur-métiers à l'origine des schémas (N.B. : les schémas soumis à confidentialité sont identifiés et écartés par leurs auteurs lors de leur création ; ils n'interviennent donc pas dans ce processus).

Le principe de la recommandation est basé sur une interface Homme-Machine auprès de laquelle l'acteur-métier saisit une expression en langage naturel ou des mots-clés ; le système retourne une liste de schémas pertinents. Ces derniers sont caractérisés par un nom, un commentaire et des mots-clés. Dans le cas où l'acteur-métier n'est pas satisfait de la réponse du système, il poursuit le dialogue. La base de connaissances contient les associations entre les schémas et les requêtes décisionnelles qui leurs ont été appliquées.

Avant de présenter le processus de recommandation, nous définissons les métadonnées utilisées.

4.3.1. Définition des métadonnées

Le schéma de l'entrepôt (appelé schéma global) est établi lors de la création de l'entrepôt (hebdomadaire pour notre cas d'étude). Le schéma global est composé d'un ensemble de classes contenant des objets (aussi appelés enregistrements, tuples, documents). Une classe est définie par l'ensemble des attributs qui constituent ses objets. Chaque attribut est associé à un type de donnée qui peut être atomique (entier, réel, chaîne de caractères, etc.) ou complexe, c'est-à-dire élaboré grâce à des constructeurs (structures, ensembles, listes). Les liens d'association entre les objets sont exprimés à travers des attributs contenant des adresses d'objet (références vers d'autres objets). Les liens de composition et d'héritage sont représentés par des structures particulières.

$$Soit\ DW : \{C_1, C_2, \ldots, C_n\},\ \forall i \in [1..n],\ C_i : \{(A_1^i, A_2^i, \ldots, A_p^i)\}$$

La fonction *A_Type* appliquée à un nom d'attribut retourne son type de donnée. Un type spécifie l'ensemble des valeurs et des opérations possibles pour un attribut. Dans le cas d'un attribut-référence, le type qui lui est associé correspond au nom d'une classe (domaine de définition) présente dans le DW.

Le dictionnaire des attributs est élaboré de façon concomitante à la création du schéma global. Il répertorie les attributs du DW (au premier niveau de chaque classe). Ainsi, pour une classe Employés dont l'Adresse contient une Rue et une Ville, c'est le nom de l'attribut Employes.Adresse qui est seul répertorié. En effet, le niveau d'accès aux constituants n'est pas significatif dans notre processus de recommandation. Ce dictionnaire noté DA se présente donc sous la forme d'une liste d'attributs ; chacun d'eux est caractérisé par un numéro d'ordre, le nom d'une classe et le nom de l'attribut.

DA = $\{D_k\}$ où k ∈ ℕ* est un n° d'ordre

∀ D_k ∈ DA, D_k = (C, A) avec C ∈ DW et A ∈ C

Card(DA) = $\sum_{i=1}^{n}$ Card(C_i) avec C_i ∈ DW et n = Card(DW).

Les métadonnées concernent également les magasins qui ont été élaborés par l'ensemble des acteur-métiers du système et qui pourront faire l'objet de recommandations. Les contenus des magasins sont conservés entre deux alimentations de l'entrepôt, ce qui évite leur rechargement en cas de réutilisation. Mais lors de la création (hebdomadaire) d'une nouvelle version

de l'entrepôt, les contenus de magasin antérieurs sont supprimés en raison de leur obsolescence possible ; seules les métadonnées de ces magasins sont préservées et mises à jour pour les rendre cohérentes avec le schéma global. Notons que pour assister les acteur-métiers dans la gestion des schémas, le processus DWtoDM demande que tout magasin créé soit accompagné de métadonnées telles qu'un nom, un commentaire et des mots-clés. Un magasin est extrait de l'entrepôt ; son schéma est donc une restriction de celui de l'entrepôt et se définit dans les mêmes termes.

L'historique des sessions est également constitué pour conserver les données manipulées par l'acteur-métier. Ainsi, à la fin de chaque session, le module de requêtage alimente un fichier « QLog » contenant la liste des requêtes. Si U représente l'ensemble des acteur-métiers du système, alors le fichier $QLog^u$ est défini comme un ensemble de sessions :

$$QLog^x = \{S_i^u\} \text{ où i} \in \mathbb{N}^* \text{ est un n° d'ordre et u} \in \text{U.}$$

Chaque session contient un ensemble de requêtes mises en œuvre par l'acteur-métier.

$$\forall\ S_i^u \in QLog^x, S_i^u = \{R_1, R_2, \ldots, R_n\}$$

où n est le nombre de requêtes émises par l'acteur-métier u pendant la session.

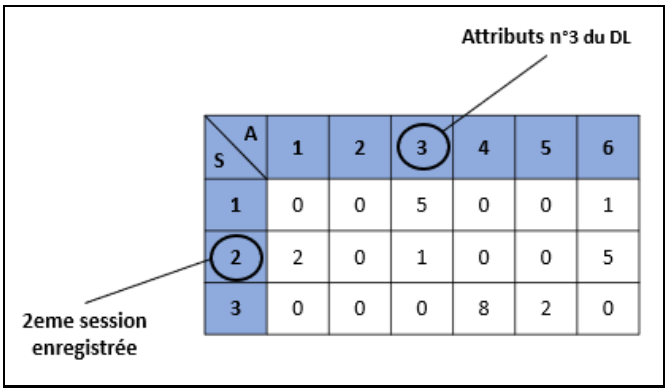
Une session étant associée à un magasin unique, toute requête de type SQL est exprimée à partir du schéma décrivant ce magasin. Si nous excluons les opérateurs et les valeurs de sélection, une requête contient (1) un ensemble d'attributs-résultats (clause SELECT), (2) un ensemble de classes (FROM) et (3) un ensemble d'attributs-prédicats (WHERE). A partir du fichier QLog, le processus DWtoDM alimente un historique synthétique des sessions. Cet historique fournit, pour chaque session, l'ensemble des attributs de premier niveau utilisés par les requêtes. Il utilise notamment la fonction *QAtt(s)* qui retourne l'ensemble des attributs d'une session *s*, c'est-à-dire l'union des attributs-résultats et attributs-prédicats apparaissant dans les différentes requêtes de s. De plus, le processus calcule l'effectif de chaque attribut dans une session et peut ainsi constituer un vecteur pour la session s. Soit t le nombre de requêtes dans *s*, le vecteur d'une session *s* est défini par un ensemble de couples (attribut, effectif) :

$$VS_s = \{(A^1, f^1), A^2, f^2), \ldots A^t, f^t)\}$$

$\forall$ i $\in$ [1..t], $A^i \in QAtt(s)$ et f_i représente le nombre d'apparitions de A^i dans l'ensemble des requêtes de s.

La matrice Historique-session (HS) est constituée à partir des vecteurs-sessions enregistrés sur une période passée et donne l'usage des attributs

manipulés dans les sessions. HS a pour dimension (x, y) où x représente le nombre de sessions enregistrées dans l'historique et y correspond au nombre exhaustif d'attributs du LD. Ainsi, HS contient les effectifs des attributs (de premier niveau) pour les différentes sessions enregistrées. Notons que, pour un attribut du LD (colonne) qui n'est pas utilisé dans une session, la ligne correspondante à la session contiendra l'effectif 0. La figure 4.6 montre un extrait d'une matrice HS.



S \ A	1	2	3	4	5	6
1	0	0	5	0	0	1
2	2	0	1	0	0	5
3	0	0	0	8	2	0

Figure 4.6 – Exemple partiel d'une matrice HS contenant les effectifs des attributs par session

Lors de la création d'une nouvelle version de l'entrepôt, la matrice HS est mise à jour à partir du dictionnaire des attributs. Les attributs absents du dictionnaire sont supprimés ; les nouveaux attributs sont ajoutés et leurs colonnes mises à 0 dans HS. Il est à noter que les numéros d'ordre des attributs supprimés ne sont pas réaffectés.

La section suivante présente le processus qui va permettre au processus DWtoDM de proposer des schémas de magasin selon les besoins exprimés par les acteurs-métier.

4.3.2. Processus de recommandation

Un acteur-métier qui déclenche une session, doit utiliser le schéma de magasin le plus adapté aux données qu'il souhaite explorer. Il peut créer son propre schéma mais cette opération peut s'avérer fastidieuse compte tenu de la complexité des structures de données. Pour pallier cette difficulté, le processus DWtoDM dispose d'un mécanisme de recommandation qui est basé sur un principe largement utilisé dans le domaine de la recherche d'information du Web. De façon générale, la recommandation consiste à proposer des entités à un acteur-métier après les avoir filtrées en fonction du comportement passé d'autres acteur-métiers. Ces entités peuvent être indifféremment des produits, des personnes, des photos, des magasins ; dans notre cas, il s'agit de schémas de magasins. On distingue trois types de

mécanismes de recommandation : la recommandation basée sur le contenu des entités, celle basée sur le filtrage collaboratif et celle qui combine les deux techniques. Nous avons opté pour le filtrage collaboratif qui permet de prendre en compte les expériences des autres usagers. Ainsi, on suppose que deux acteur-métiers manipulant des données voisines présentent le même comportement et donc partagent les mêmes centres d'intérêt. On considère qu'un acteur-métier exprime son besoin auprès du système et, en retour, reçoit des schémas existants adaptés à ce besoin. Rappelons que les schémas jugés confidentiels par leurs auteurs sont écartés de ce processus.

4.3.2.1- Expression du besoin

Un acteur-métier exprime son besoin par une ou plusieurs phrases en format libre correspondant généralement à une thématique. Par exemple « Obtenir la liste des consultations de médecins généralistes liées au COVID en France au dernier trimestre 2022 » ou bien « Obtenir la liste des accidents du travail dans les entreprises de moins de 80 salariés ». En retour, le système va recommander à l'acteur-métier des schémas de magasin adaptés à son besoin.

Le besoin de l'acteur-métier est alors converti en un ensemble d'attributs de la BD par le module NLP (Natural Language Processing). Comme le montre la figure 4.7, le Transformateur élimine les terminaisons, les mots-outils et les signes de ponctuation de cette chaîne de caractères ; Il obtient ainsi un ensemble de mots pertinents qui, soumis au Mappeur, vont être associés d'abord à des synonymes puis, une fois complétés, à des attributs de la BD. Le Mappeur utilise d'une part une ontologie de synonymes élaborée par des experts et qui fait correspondre un mot à des mots voisins et d'autre part une base de correspondance entre un mot et des attributs de la BD. Ces deux bases de connaissances ne sont évidemment pas exhaustives, ni concernant les mots ni concernant les attributs. Par exemple, le mot cardiaque est lié (entre autres) aux mots cœur, coronaire, infarctus ; le mot cœur est associé à des attributs de l'entrepôt tels que Pression_Art, Frequence_Car, SpO2. Notons que les attributs sont ceux présents dans le dictionnaire des attributs. Les ontologies peuvent être complétées par les experts s'ils souhaitent améliorer l'efficacité du système ; mais l'ontologie de correspondance mots/attributs est systématiquement mise à jour lors de l'évolution du dictionnaire des attributs par la suppression des attributs non répertoriés. Un extrait de l'ontologie de synonymes est donné dans le chapitre 5. De plus, le module Mappeur calcule le poids d'un attribut, c'est-à-dire le nombre de fois où un attribut est associé au besoin de l'acteur-métier. Ainsi, pour un besoin exprimé par l'acteur-métier, le Mappeur produit un vecteur VB de la forme suivante :

VB = $\{(A^1, w^1), (A^2, w^2), \ldots \{(A^q, w^q)\}$ où q représente le nombre d'attributs déduits du besoin.

$\forall$ i $\in$ [1..q], A^i est un attribut associé au besoin et w^i représente son poids (nombre d'apparitions de A^i dans le besoin).

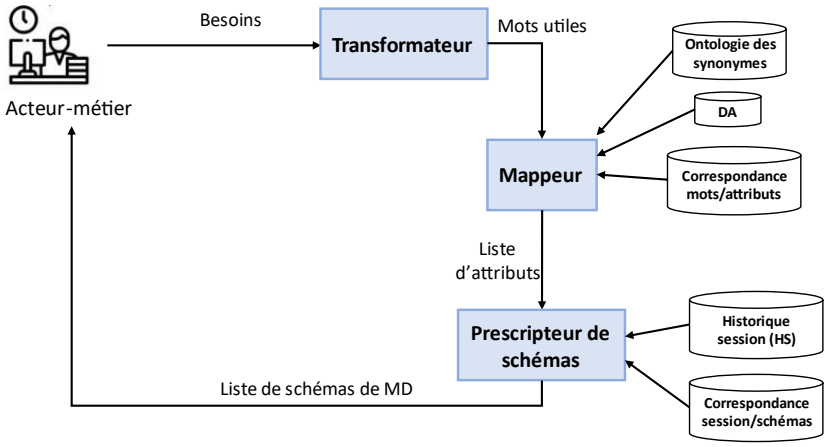


Figure 4.7 – Architecture du mécanisme de recommandation

Nous présentons ci-après un exemple concret pour représenter l'étape de l'expression du besoin dans un contexte médical. Un médecin souhaite obtenir les données relatives aux patients atteints de diabète et examiner les facteurs qui influencent leur taux de glucose. Le médecin veut explorer les relations entre l'âge, l'indice de masse corporelle (IMC), la tension artérielle et le taux de glucose chez les patients diabétiques.

Étape 1 : Expression du besoin

Le médecin exprime son besoin sous la forme d'une phrase : "Relations entre l'âge, l'indice de masse corporelle, la tension artérielle et le taux de glucose dans le sang chez les patients diabétiques et les patients hypertendus âgés de plus de 50 ans".

Étape 2 : Conversion en vecteur du besoin (VB)

Les modules Transformateur et Mappeur présentés dans la figure 4.8 reposent sur des techniques de NLP (Natural Language Processing) [89] notamment la suppression des mots-outils et les ponctuations et traitent cette phrase en effectuant les étapes suivantes :

Élimination des terminaisons, des mots-outils et des signes de ponctuation :

"Relations âge indice de masse corporelle tension artérielle taux de glucose sang patients diabétiques patients hypertendus âgés plus 50 ans"

Obtention d'un ensemble de mots pertinents :

{"âge", "indice de masse corporelle", "tension artérielle", "taux de glucose", "patients diabétiques", "hypertendus", "plus 50 ans"}.

Ces mots pertinents sont ensuite soumis au Mappeur (cf. Figure 4.7), qui associe chaque mot à des synonymes et à des attributs de la base de données médicale (entrepôt). Le Mappeur utilise une ontologie élaborée par des experts, ainsi qu'une base de correspondance entre les mots et les attributs de la BD. Par exemple :

- le mot "âge" est associé à l'attribut "Patients.Age"
- le mot "indice de masse corporelle" est associé à l'attribut "Patients.IMC"
- le mot "tension artérielle" est associée à l'attribut "Patients.Pression_Art"
- le mot "taux de glucose" est associé à l'attribut "Patients.Glycemie"
- le mot "patients diabétiques" est associé à l'attribut "Patients.Diabetique"
- le mot "patients hypertendus" est associé à l'attribut "Patients.Pression_Art"
- le mot "50 ans" peut être associé à l'attribut "Patients.Age"

Le Mappeur complète ainsi l'ensemble des mots pertinents avec les attributs correspondants de la BD. Ce processus permet de convertir le besoin de l'acteur-métier en un ensemble structuré d'attributs de la base de données, présenté sous forme de vecteur VB qui servira de base pour la recommandation de schémas de magasin adaptés à ce besoin.

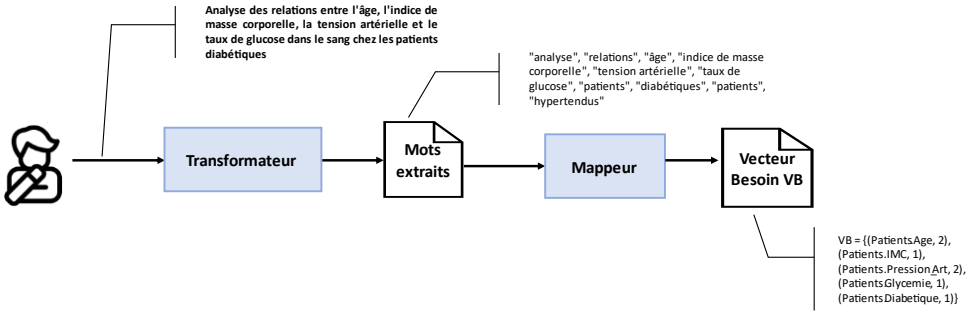


Figure 4.8 – Exemple du fonctionnement des modules Transformateur et Mappeur dans un cadre médical

Dans cet exemple, le vecteur VB est composé de 5 paires (A, w), où A représente un attribut de la base de données déduit du besoin de l'acteur-métier, et w représente son poids, c'est-à-dire le nombre de fois où cet attribut apparaît dans le besoin exprimé. Ainsi, le poids (w) associé à l'attribut Patients.Age est de 2, ce qui signifie que l'attribut est utilisé deux fois dans le besoin.

VB = {(Patients.Age, 2), (Patients.IMC, 1), (Patients.Pression_Art, 2), (Patients.Glycemie, 1), (Patients.Diabetique, 1)}

Ces associations entre les mots clés et les attributs permettent au système de recommandation de générer des schémas de magasin pertinents pour obtenir des données médicales liées à l'âge, à l'IMC, à la pression artérielle, à la glycémie et aux patients diabétiques et hypertendus. Ainsi, l'étape d'expression du besoin permet de convertir la requête en langage naturel de l'acteur-métier en un ensemble d'attributs de la BD. Ceci est représenté sous la forme d'un vecteur VB qui sera utilisé par la suite dans le processus de recommandation pour trouver les schémas de magasin pertinents.

4.3.2.2- Prédiction de schémas

Notre mécanisme de recommandation est basé sur le filtrage collaboratif. Le filtrage est passif dans la mesure où les acteur-métiers ne donnent pas explicitement une évaluation (notation) des magasins utilisés dans le passé. Le modèle utilisé prédit les schémas de magasin les plus proches du besoin exprimé par l'acteur-métier et leur attribue un score d'adéquation. Le module Prescripteur (cf. Figure 4.7) reçoit en entrée un vecteur du besoin VB qui contient la liste des attributs déduits. Il dispose aussi des métadonnées telles que la matrice HS contenant l'historique des sessions passées et le dictionnaire des attributs DA. L'objectif de Prescripteur est d'affecter automatiquement un score aux sessions en mettant en regard les attributs du besoin (VB) et les attributs manipulés par chaque session (HS).

Rappelons que la matrice HS contient un ensemble de sessions du passé déclenchées par les acteur-métiers du système et qui ont permis d'explorer les entrepôts sur une période de temps. Lors de l'ouverture d'une session par un acteur-métier, le module Prescripteur créé une matrice de travail WHS associée à la session et qui contient l'intégralité des coefficients de HS.

Reprenons les éléments de la section précédente pour illustrer le fonctionnement du module Prescripteur :

Vecteur du besoin VB :

VB = {(Patients.Age, 2), (Patients.IMC, 1), (Patients.Pression_Art, 2), (Patients.Glycemie, 1), (Patients.Diabetique, 1)}

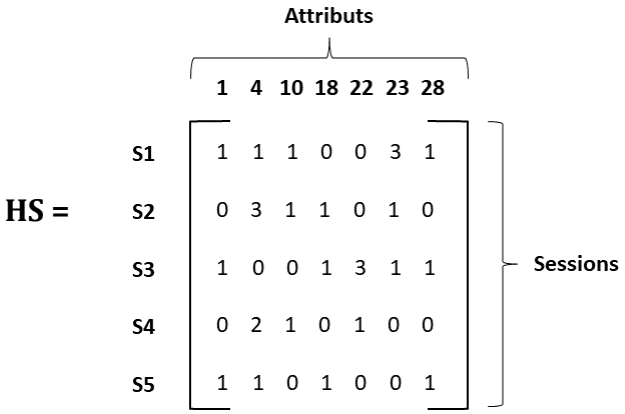
Dictionnaire des attributs DA :

N° d'ordre	Nom de l'attribut	Type de l'attribut	Libellé de l'attribut	Classe source
1	Patients.NSS	Chaîne de caractères	Identifiant du patient (Numéro de sécurité sociale)	Patients

4	Patients.Age	Entier	Age du patient	Patients
10	Patients.Medicaments	Référence	Liste des médicaments du patients	Patients
18	Patients.IMC	Nombre réel	Indice de masse corporelle du patient	Patients
22	Patients.Pression_Art	Nombre réel	Mesure de la pression artérielle du patient à un instant t	Patients
23	Patients.Glycemie	Nombre réel	Mesure de la glycémie du patient à un instant t	Patients
28	Patients.Diabetique	Boolean	Indication si patient diabétique ou non diabétique	Patients

Tableau 4.1 – Extrait du dictionnaire de données DA

La matrice des sessions HS :



La matrice HS suivante représente l'historique des sessions passées. La valeur (S3,22) dans la matrice indique que l'attribut 22 a été manipulé 3 fois dans la session 3.

Lors de l'ouverture d'une nouvelle session par un acteur-métier, le module Prescripteur crée une matrice de travail WHS initialisée avec HS.

4.3.2.3- Pondération des attributs

D'une part, le vecteur VB est élaboré au début de la session courante ; il contient les poids des attributs associés au besoin d'un acteur-métier. D'autre part, les lignes de la matrice WHS contiennent les effectifs des attributs manipulés par les sessions du passé. Lors de cette étape, le module Prescripteur reporte le poids des attributs du besoin dans la matrice WHS.

Soit x la session courante et soit la fonction *PoidsVB(x,y)* qui retourne le poids de l'attribut y présent dans le vecteur du besoin VB_x, pour tout i = 1..n où n représente le nombre de sessions enregistrées dans HS, pour tout j = 1..q où q représente le nombre d'attributs dans VB_x,

$$WHS_{i,j} = HS_{i,j} * PoidsVB(x,j).$$

Cette opération effectue le produit de l'effectif d'un attribut dans le déroulement d'une session avec le poids de ce même attribut dans VB. Elle permet ainsi de marquer l'importance d'un attribut du besoin dans les différentes sessions qui l'ont utilisé.

Reprenons l'exemple médical pour illustrer la pondération des attributs dans le contexte de recommandation de schémas de magasins.

Nous allons pondérer les attributs de la matrice HS en utilisant les poids du vecteur VB (cf. figure 4.9).

Poids de VB : [2, 1, 2, 1, 1]

Matrice des sessions pondérée WHS : WHS = HS x VB

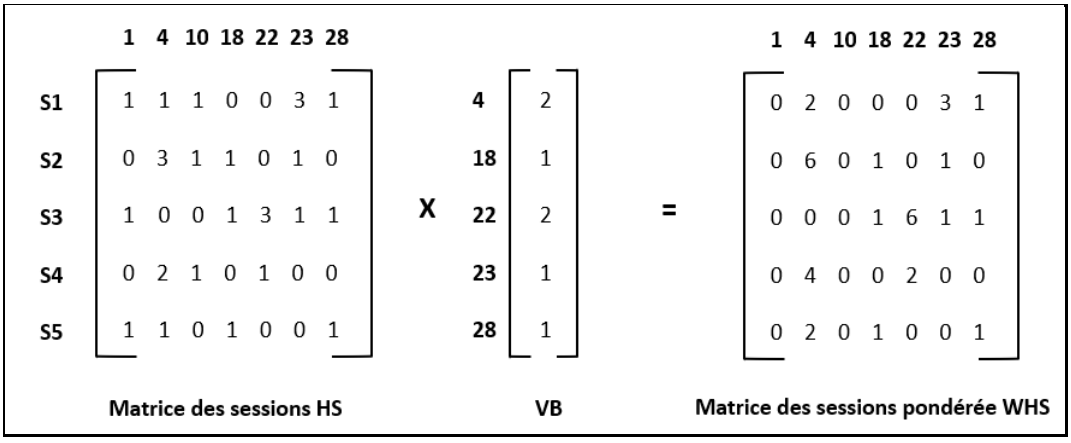


Figure 4.9 - Pondération des attributs dans la matrice WHS

4.3.2.4- Restriction de la matrice WHS

A ce stade, la matrice WHS dispose des colonnes correspondantes à tous les attributs de DA ; or, la totalité de ces attributs n'apparaissent pas dans VB. Pour éliminer les attributs sans intérêt pour la session courante, le module Prescripteur crée la sous-matrice RWHS en réduisant la matrice WHS aux seuls attributs de VB.

Soit la matrice WHS de dimension (n, p) où n et p correspondent respectivement au nombre des sessions enregistrées et au nombre d'attributs dans DA et soit la fonction *AttVB(x)* qui retourne l'ensemble des numéros d'ordre des attributs associés au besoin du vecteur VB_x ; soit q la cardinalité de cet ensemble

Pour tout i = 1..n où i représente le numéro d'ordre d'une session dans HS et pour tout j = 1..q où *j* représente un numéro d'ordre des attributs du besoin VB_x :

$$RWHS_{i,j} = WHS_{i,j} \quad avec\ j \in AttVB(x).$$

La matrice WHS (Working Hypotheses Schema) représente les sessions pondérées, où les attributs ont été ajustés en fonction des poids du vecteur VB. La matrice WHS est restreinte en ne conservant que les sessions contenant les attributs pondérés. Par exemple, si une session ne contient pas les attributs "Patients.Diabétique" et "Patients.Pression_Artérielle", elle sera exclue de la matrice. Cela permet de donner plus de poids aux attributs importants selon le besoin exprimé par l'acteur-métier. Le module Prescripteur crée une sous-matrice RWHS en restreignant la matrice WHS aux attributs présents dans le vecteur du besoin VB. Ceci permet de filtrer les attributs non pertinents. Dans notre exemple, la sous-matrice RWHS sera (cf. figure 4.10) :

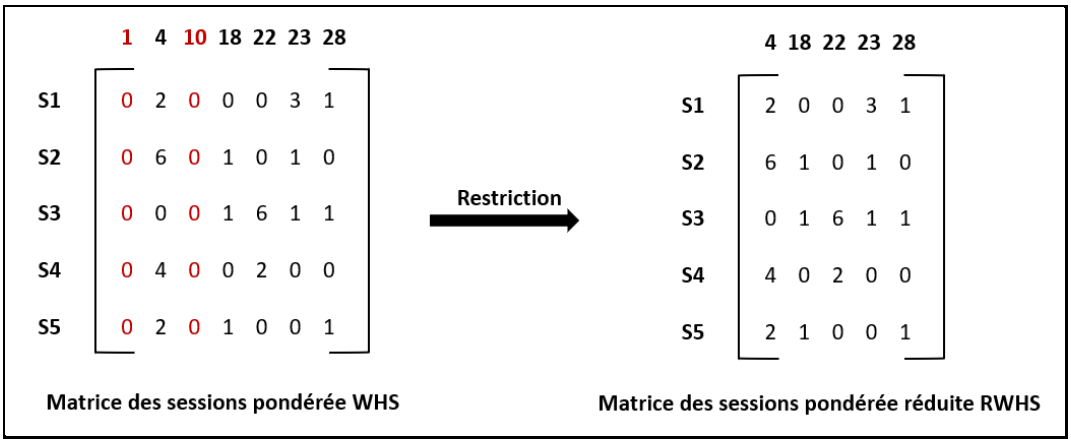


Figure 4.10 – Restriction de la matrice WHS

4.3.2.5- Pénalisation des attributs

Dans une session, certains attributs du magasin n'ont pas été manipulés par les requêtes de la session ; ce qui signifie que le schéma du magasin en question pouvait couvrir un besoin plus large. Pour que le score associé à un schéma de magasin soit plus significatif, le module Prescripteur affecte une pénalité à tous les attributs n'ayant pas été manipulés dans une session (coefficient à 0). Dans notre cas d'étude, une pénalité de -1 a été choisie à partir des simulations que nous avons effectuées.

Pour tout i = 1..n où i identifie le numéro d'ordre d'une session dans HS et pour tout j = 1..q où j représente un attribut du besoin VB :

si $RWHS_{ij} = 0$ *alors* $RWHS_{ij} = -1$ (cf. figure 4.11).

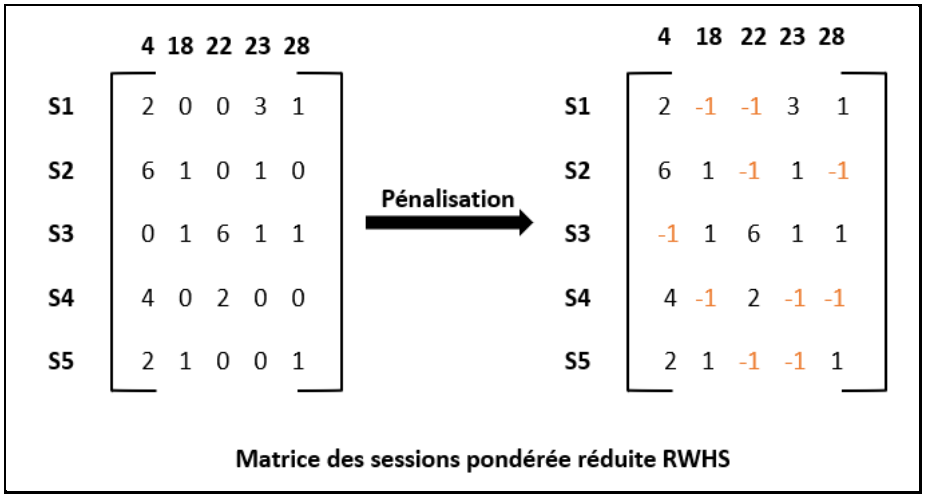


Figure 4.11 – Pénalisation des attributs de la matrice WHS

4.3.2.6- Calcul du score

En cumulant les effectifs pondérés des attributs utilisés par une session (une ligne de RWHS), nous obtenons le score de la session au regard du besoin VB de l'acteur-métier. Chaque session étant associée à un schéma de magasin unique, un score mesure donc l'adéquation du besoin avec ce schéma. De plus, dans le cas où plusieurs sessions utilisent un même schéma, le score affecté au schéma est égal au maximum des scores obtenus. L'ensemble des résultats est enregistré dans le vecteur VDM.

Dans notre exemple, les scores sont calculés pour chaque session de la matrice RWHS en sommant les valeurs des attributs dans la session :

- S1 : Score = 2 - 1 -1 + 3 + 1 = 4
- S2 : Score = 6 + 1 - 1 + 1 - 1 = 6
- S3 : Score = -1 + 1 + 6 + 1 + 1 = 8

- S4 : Score = 4 - 1 + 2 - 1 - 1 = 3
- S5 : Score = 2 + 1 - 1 - 1 + 1 = 2

Ces scores sont reportés dans le vecteur des schémas de magasin prédits (VDM) : [4, 6, 8, 3, 2].

Le score d'adéquation représente la similitude entre le besoin exprimé par l'acteur-métier et une session passée. Les sessions du vecteur VDM seront proposées à l'acteur-métier selon leurs scores dans l'ordre décroissant. Ici, les sessions S3 et S2 sont plus susceptibles de répondre au besoin de l'acteur-métier. L'algorithme 4 présente les traitements réalisés par le module Prescripteur.

Ainsi, grâce au module NLP et au module Mappeur, le mécanisme de recommandation est capable de comprendre et d'extraire les attributs pertinents à partir du besoin exprimé par l'acteur-métier.

Si les schémas proposés initialement par le mécanisme de recommandation ne sont pas adaptés, l'acteur-métier peut faire appel à des techniques vues dans la section 4.1. Il a ainsi la possibilité de personnaliser un schéma recommandé qui répond mieux à ses besoins.

Ce processus permet d'ajuster et d'affiner le schéma en fonction des besoins et des préférences individuelles de l'acteur-métier. Ainsi, l'acteur-métier a un contrôle total sur la conception et la personnalisation de son schéma du magasin, ce qui garantit une adéquation avec ses exigences.

Algorithm 4 : *Prescribe*

Input : VB : Vector of Needs, HS : Session History Matrix
Output : VDM : Vector of Predicted DM Schemas

Create the working matrix WHS

```
1   ForEach attribute A in VB do
2       ForEach session in WHS do
3           Weight the frequencies of WHS by the weights of VB
4       EndFor
5   EndFor
6   ForEach attribute in VB do
7       Create the sub-matrix RWHS by restricting the WHS matrix
8   EndFor
9   ForEach attribute in RWHS do
10      ForEach session in RWHS do
```

```
11          Penalize unused attributes in the sub-matrix RWHS
12      EndFor
13  EndFor
14  ForEach session in RWHS do
15      Calculate the score and report it in VDM
16  endFor
```

4.4. Bilan du processus de sélection

4.4.1. Synthèse

Ce chapitre propose le processus DWtoDM qui facilite la création de magasin à partir d'un entrepôt, soit par un mécanisme de personnalisation, soit par un mécanisme de recommandation. L'acteur-métier a la possibilité de reprendre un schéma existant ou de l'adapter à des besoins spécifiques (personnalisation). Par ailleurs, il a la possibilité de faire appel à un mécanisme de recommandation pour obtenir des schémas de magasin élaborés par des acteur-métiers ayant des profils similaires.

Le mécanisme de recommandation comporte plusieurs étapes. Tout d'abord, les besoins exprimés en langage naturel par l'acteur-métier sont analysés et convertis en attributs présents dans l'entrepôt. Ensuite, une pondération des attributs est effectuée selon leur présence dans le schéma. Enfin, le calcul du score final permet de classer les schémas de magasin recommandés en fonction de leur pertinence par rapport aux besoins.

La mise en œuvre du processus DWtoDM est présentée dans le chapitre 5 « Expérimentation et Validation ».

4.4.2. Positionnement

A présent, nous effectuons le positionnement de nos travaux au regard des travaux présentés dans le chapitre 2.

QueRIE [73] est un système de recommandation de requêtes SQL qui se base sur les sessions de requêtes enregistrées. Il utilise la factorisation matricielle pour identifier les similitudes entre les sessions de requêtes, ce qui permet de recommander des requêtes qui sont susceptibles d'intéresser les utilisateurs en se basant sur les préférences des utilisateurs similaires. QueRIE est un système très performant mais il nécessite un grand nombre de sessions de requêtes pour être efficace. Il peut donc être difficile à utiliser dans les cas où les utilisateurs ne sont pas très actifs.

YmalDB [74] est un système de recommandation de requêtes SQL qui vise à faciliter l'exploration des BD relationnelles. Il analyse les requêtes précédemment exécutées ainsi que les résultats obtenus pour créer un modèle de recommandation capable de suggérer des requêtes similaires qui peuvent être utiles à l'utilisateur. Toutefois, cette méthode nécessite une certaine connaissance des requêtes SQL pour être efficace. Les utilisateurs doivent être en mesure de comprendre les résultats des requêtes précédentes pour tirer profit des recommandations de YmalDB.

SnipSuggest [75] est un système d'auto-complétions de requêtes SQL qui exploite l'historique des requêtes précédentes pour identifier des corrélations entre les différentes clauses SQL fréquemment utilisées. Ces corrélations sont ensuite utilisées pour générer des extraits qui peuvent être pertinents pour la requête en cours de saisie. SnipSuggest est un système efficace pour aider les utilisateurs à formuler des requêtes SQL plus rapidement et plus facilement. Il permet aux utilisateurs de gagner du temps et de réduire les erreurs de frappe. Cependant, il peut être difficile à utiliser dans les cas où les requêtes SQL sont complexes ou non standard. Le système peut ne pas être en mesure de générer des extraits pertinents pour ce type de requêtes.

Les auteurs de [78] proposent une approche de recommandation. Des algorithmes de similarité sont utilisés pour identifier les intérêts des utilisateurs en analysant leurs historiques de requêtes. Cela permet de détecter les intentions de chaque utilisateur en lui recommandant des requêtes. Toutefois et en l'absence d'interactions avec les utilisateurs, leurs intérêts peuvent évoluer au fil du temps et cette approche peut avoir du mal à prendre en compte ces changements. Par exemple, un utilisateur qui avait l'habitude de rechercher des informations sur un sujet spécifique peut changer d'intérêt ; le système de recommandation ne prendra pas en compte cette évolution dont il ignore l'existence.

L'article [77] recommande des requêtes à un utilisateur lorsqu'il saisit des mots-clés dans un moteur de recherche. Les recommandations sont basées sur l'historique de recherche des utilisateurs, l'analyse sémantique des mots-clés et la compréhension de l'intention de recherche de l'utilisateur. Ainsi, en tenant compte des préférences et des comportements de recherche individuels, le système peut fournir des recommandations plus adaptées à chaque utilisateur.

Nous venons de présenter un panorama des principaux travaux cités dans l'état de l'art et développés en vue de faciliter l'accès à des données complexes. Nous avons pu constater que ces solutions consistent principalement à recommander à un utilisateur des requêtes élaborées par d'autres utilisateurs dont les comportements sont proches. Nos travaux se situent dans la même veine ; ils doivent assister des acteurs-métier dans l'interrogation d'un entrepôt contenant des données aux thématiques

multiples et présentant des structures complexes. Chaque acteur-métier (ou classe d'acteurs-métier), ayant des besoins spécifiques en matière de données à manipuler, nous avons développé un système de recommandation de magasins, c'est-à-dire de schémas extraits de l'entrepôt. Les magasins simplifient grandement la rédaction des requêtes dans la mesure où leurs contenus sont limités aux besoins en données exprimés par les acteurs-métier.

CHAPITRE 5

EXPÉRIMENTATION ET VALIDATION

Nos travaux s'inscrivent dans le cadre d'une convention CIFRE où l'aspect mise en œuvre des concepts constitue une facette importante. L'entreprise TRIMANE a des attentes précises pour l'amélioration de certains processus qu'elle développe pour le compte de ses clients. Ce chapitre est consacré à l'implantation d'un prototype qui apportera d'une part des principes et des démarches dont les ingénieurs pourront s'inspirer et d'autre part des modules logiciels qui pourront être directement implantés dans des applications clientes.

L'objectif de nos travaux est de faciliter l'accès à des bases hétérogènes contenues dans un LD mis à la disposition d'acteurs-métier. En effet, aussi bien au niveau industriel que dans les travaux de recherche, les entrepôt et magasin sont conçus et créés par des experts en BI pour le compte des acteurs-métier. Afin de montrer la pertinence de nos propositions, nous avons développé le système DLtoDM, un logiciel mettant en œuvre les principes et techniques proposés dans notre thèse.

Le présent chapitre décrit le prototype DLtoDM, notamment son architecture et les interfaces destinées aux acteurs-métier. Ce chapitre permet de vérifier et d'expérimenter nos propositions présentées dans les chapitres 3 et 4 à travers une mise en œuvre concrète et une évaluation effectuée dans une entreprise, en mettant en avant la faisabilité et l'efficacité de notre prototype. Cette évaluation porte essentiellement sur la facilité d'utilisation de notre logiciel par des non informaticiens.

Le chapitre 5 est structuré de la manière suivante. La section 5.1 décrit l'architecture technique de notre système. La section 5.2 décrit le jeu de données utilisé pour expérimenter DLtoDM. La section 5.3 décrit le prototype DLtoDW permettant de créer un entrepôt à partir d'un LD ainsi que les outils techniques utilisés pour l'implanter. La section 5.4 présente le processus DWtoDM avec ses deux mécanismes de personnalisation et de recommandation. Enfin, la section 5.5 propose une évaluation de notre système.

5.1. Architecture technique de DLtoDM

Un utilisateur du système DLtoDM est un acteur-métier qui souhaite accéder aux données d'un LD. Pour ce faire, le système offre deux fonctions :

- la construction d'un entrepôt NoSQL à partir du LD,
- la création du schéma d'un magasin basé sur l'entrepôt.

La Figure 5.1 montre les deux processus de DLtoDM correspondants à chacune de ces fonctions : (1) le processus DLtoDW qui extrait les données à

partir d'un LD et crée l'entrepôt et (2) le processus DWtoDM qui crée des schémas de magasin en fonction des besoins de l'acteur-métier.

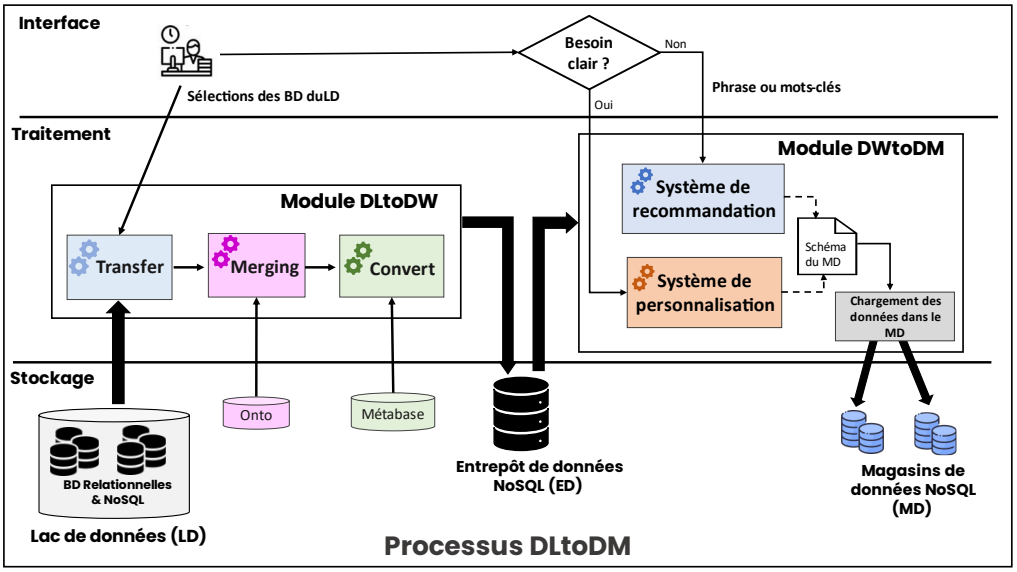


Figure 5.1 – Architecture technique du prototype DLtoDM

DLtoDM a été développé en JAVA et comporte 3 niveaux :

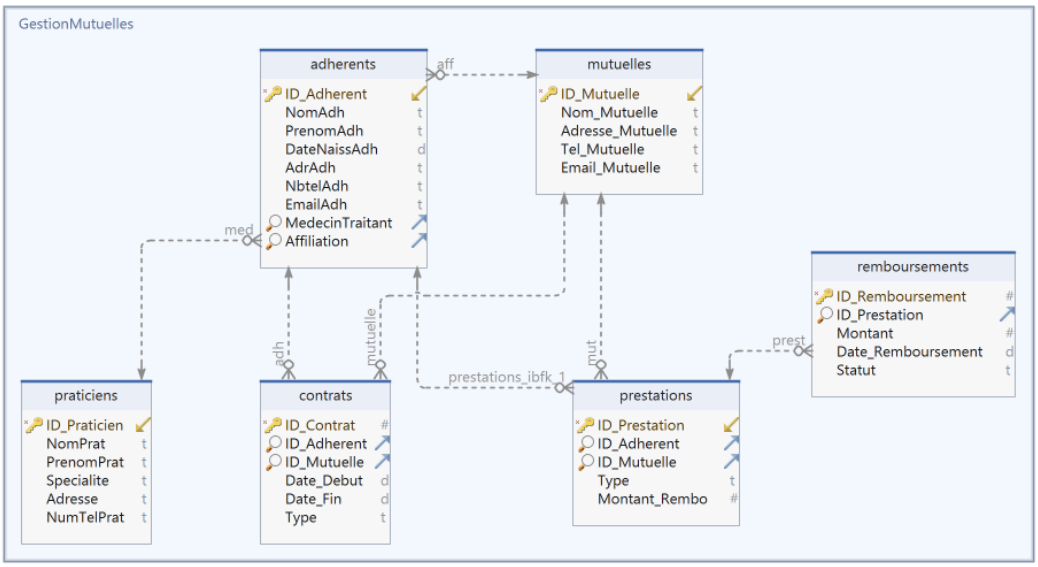
- le niveau « Stockage » qui correspond à l'ensemble des BD contenues dans le LD ainsi que les métadonnées en entrée et les sorties des différents modules (l'entrepôt et les magasin). L'ensemble des métadonnées servira au processus DLtoDW pour alimenter l'entrepôt et au processus DWtoDM pour proposer des schémas de magasin adéquats.
- le niveau « Traitement » se décompose en deux processus : (1) le processus DLtoDW qui assure le transfert des BD du LD choisies par l'utilisateur vers un entrepôt ; (2) le processus DWtoDM qui crée des magasin à partir de l'entrepôt. Ceci repose sur des interactions avec l'utilisateur.
- le niveau « Interface » qui met les différentes BD du LD ainsi que les schémas de magasin à la disposition du acteur-métier.

5.2. Jeu de données

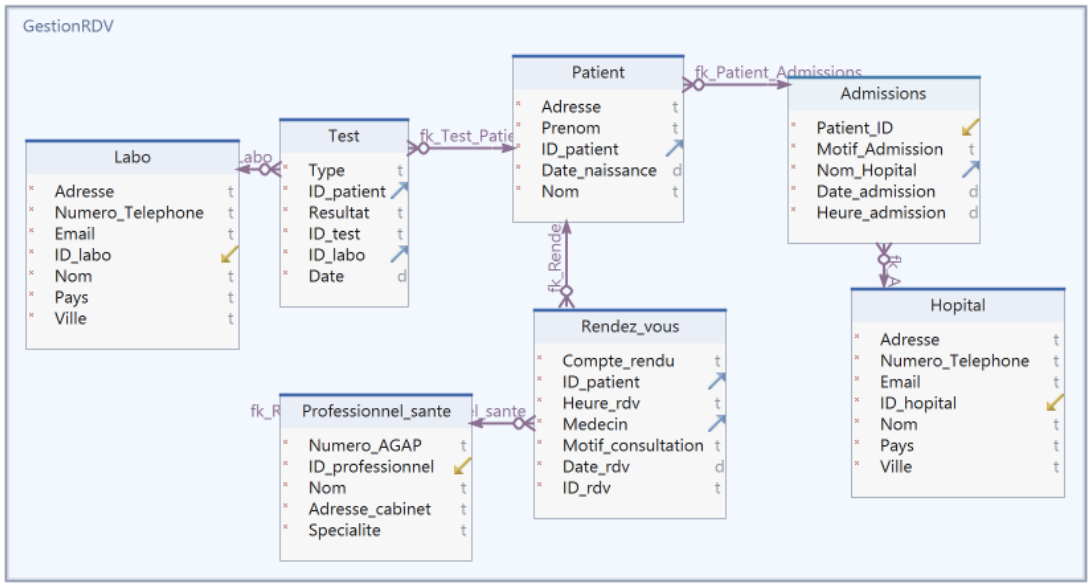
Pour illustrer le fonctionnement de DLtoDM, nous prenons le cas d'un LD contenant des données tests directement inspirées de l'étude de cas médicale réelle (cf. chapitre 1) telles que les dossiers médicaux électroniques, les

résultats de laboratoire, les rapports d'imagerie médicale, etc. Des extraits des schémas des BD composant le LD sont présentés dans la figure 5.2.

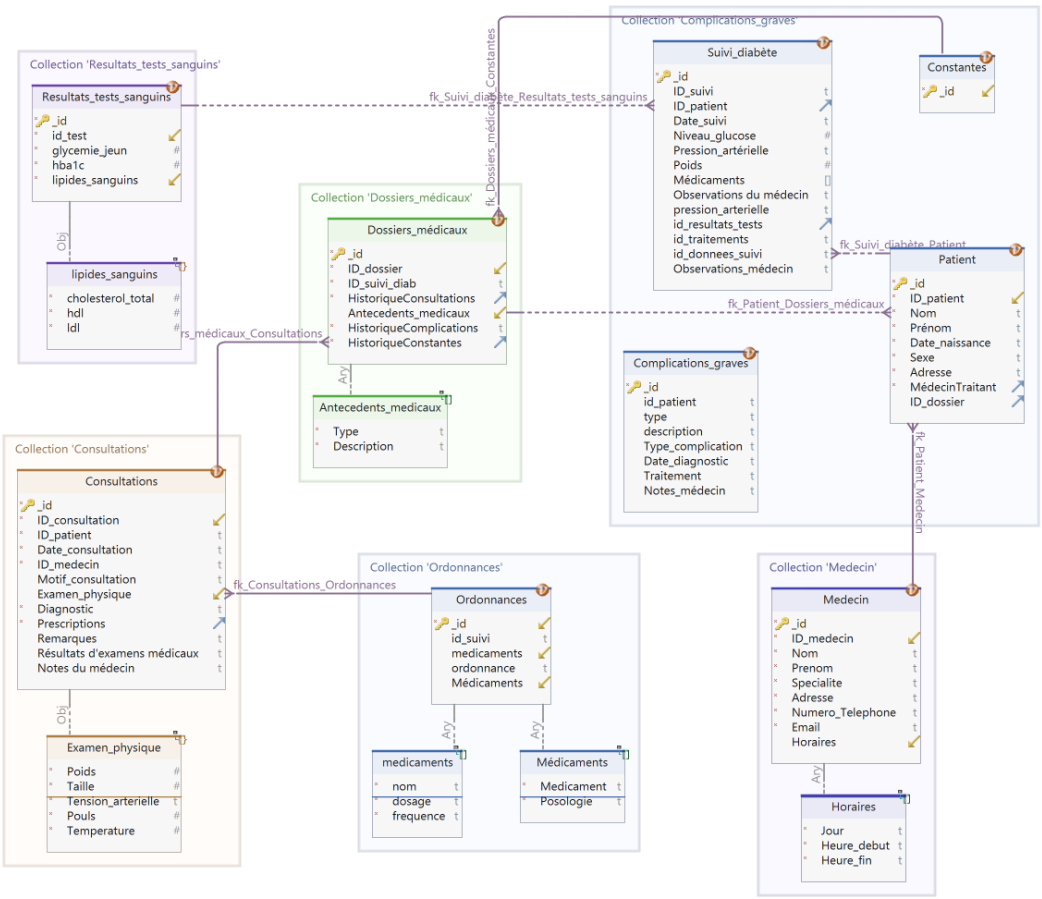
Lors du démarrage du logiciel DLtoDM, l'utilisateur s'identifie et indique la ou les BD sources qu'il souhaite interroger. Nous rappelons que, dans la version actuelle de DLtoDM, seules les sources relationnelles et NoSQL orientées-documents sont considérées ; en effet ces catégories de données représentent une partie importante des données de l'ENS sans toutefois couvrir la totalité de l'ENS.



5.2.a – BD relationnelle « GestionMutuelles »



5.2.b – BD OrientDB « GestionRDV »



5.2.c – BD MongoDB « DossierMedical »

Figure 5.2 – Extraits des schémas de trois BD du LD

Cette application gère les dossiers médicaux des patients, les suivis des traitements et des visites médicales et les informations liées à la prise en charge des dossiers dans les mutuelles. Les dossiers médicaux comprennent les antécédents médicaux, les diagnostics, les ordonnances et les résultats des examens médicaux. Chaque patient est suivi tout au long de son parcours de soins, avec des informations enregistrées à chaque visite médicale. De plus, l'application enregistre les informations sur les traitements prescrits aux patients, y compris les médicaments, les posologies et les durées de traitement. Les médecins peuvent suivre et mettre à jour les traitements en fonction de l'évolution de l'état de santé du patient. Les données provenant de différentes sources médicales, telles que les résultats de tests de laboratoire, les images médicales et les rapports d'hospitalisation, sont également intégrées dans l'application. Cela permet aux médecins d'accéder

rapidement et facilement à toutes les informations pertinentes pour prendre des décisions sur les soins à prodiguer aux patients.

Il est à noter que les noms des attributs présents dans les BD réelles ont été changés pour des raisons de confidentialité et pour faciliter la lecture des exemples.

5.3. Processus DLtoDW

L'ingestion des données du LD vers une BD cible est une étape essentielle dans la construction d'un entrepôt. Dans cette section, nous présentons le processus DLtoDW qui centralise–les données provenant du LD vers un entrepôt. Ceci permet aux acteur-métiers d'accéder facilement aux données d'un LD. Pour mettre en œuvre le processus DLtoDW, nous avons développé une architecture modulaire qui se compose de trois modules exécutés successivement. Ces modules assurent une ingestion efficace des données du LD vers l'entrepôt. Le premier module *Transfer* est responsable de la conversion des données des BD sources vers l'entrepôt. Ce module utilise des techniques proposées par l'architecture MDA pour faciliter le transfert des données tout en préservant leur intégrité. Le deuxième module *Merging* a pour objectif de fusionner les classes similaires afin de garantir une meilleure organisation des données dans l'entrepôt. Ce module utilise une ontologie fournie par des experts pour identifier et regrouper les classes similaires. Le troisième module *Convert* se charge de transformer les liens relationnels et NoSQL en références.

Dans les sections suivantes, nous détaillerons le rôle et le fonctionnement de chaque module et son rôle dans le processus global de DLtoDW.

5.3.1. Architecture technique de DLtoDW

Pour mettre en œuvre le processus DLtoDW, nous avons développé un prototype permettant l'ingestion de plusieurs BD relationnelles gérées par les systèmes MySQL[20] et PostgreSQL[21] et des BD NoSQL gérées par les systèmes MongoDB et OrientDB. La cible du processus, c'est-à-dire l'entrepôt, est une BD NoSQL supportée par une plateforme OrientDB ; elle doit permettre d'obtenir les parcours de soins des assurés atteints de pathologies chroniques. Le processus DLtoDW comporte trois niveaux (cf. sfigure 5.1) :

[20] https://www.mysql.com
[21] https://www.postgresql.org

- le niveau « Stockage » qui correspond à l'ensemble des BD contenues dans le LD ; il contient également les métamodèles des BD en entrée et de l'entrepôt.
- le niveau « Traitement » se décompose en trois modules *Transfer*, *Merging* et *Convert* qui sont mis en œuvre successivement. Nous avons choisi l'architecture MDA de l'OMG pour développer le module *Transfer*. Les deux modules *Merging* et *Convert* reposent sur des algorithmes codés en Java.
- le niveau « Interface » qui permet à l'utilisateur de choisir les BD à transférer du LD vers l'entrepôt.

5.3.2. Outils d'implantation

Dans cette section, nous présentons les techniques que nous avons utilisées pour mettre en place notre environnement expérimental. Étant donné que notre approche repose sur MDA, il est essentiel d'avoir une infrastructure adaptée à la méta-modélisation, à la modélisation et aux transformations de modèles M2M (Model-To-Model). Ainsi, nous utilisons EMF[22] (Eclipse Modeling Framework) [90] comme plateforme d'implantation. EMF offre un ensemble d'outils conçus pour faciliter le développement basé sur les modèles au sein de l'environnement Eclipse. Ces outils offrent trois fonctionnalités principales. La première permet de définir un métamodèle qui représente les sources utilisées par l'acteur-métier. La deuxième crée des modèles qui instancient ce métamodèle. Enfin, la troisième fonctionnalité concerne les transformations de modèles. Dans ce qui suit, nous allons présenter les outils d'EMF que nous avons utilisés pour mettre en œuvre notre prototype.

5.3.2.1 Ecore

Ecore[23] est un langage de méta-modélisation d'EMF. Ce langage se concentre sur la spécification structurelle des métamodèles pour créer des modèles de données [91]. Il s'agit d'une implantation d'un sous-ensemble du standard MOF (Meta Object Facility) de l'OMG. Ecore fournit une représentation formelle pour décrire les modèles de données en utilisant des concepts tels que les classes, les attributs, les associations et les opérations. En utilisant Ecore, les développeurs peuvent générer automatiquement du code Java correspondant aux modèles créés. Ceci permet de faciliter le développement d'applications basées sur des modèles.

[22] https://www.eclipse.org/modeling/emf
[23] ECore. The eclipse modeling framework project home page. http ://www.eclipse.org/emf

5.3.2.2 XMI

L'environnement EMF intègre la norme XMI (XML Metadata Interchange) pour faciliter l'échange de métadonnées entre les outils de modélisation et les systèmes. Cette norme est basée sur XML (eXtensible Markup Language) et offre une structure permettant de représenter et d'échanger des modèles de données, des schémas et d'autres informations structurelles.

Le fonctionnement de XMI repose sur la génération automatique de grammaires XML (DTD - Document Type Definition) à partir d'un métamodèle, comme le métamodèle Ecore utilisé dans notre cas. Cette approche permet de représenter les modèles instanciés sous la forme de documents XML.

XMI utilise une analogie entre les modèles et leur métamodèle d'une part, et les documents XML et leur DTD d'autre part. Cette analogie se manifeste par le fait qu'un métamodèle et une DTD définissent respectivement les structures des modèles et des documents XML. La figure 5.4 illustre le principe de fonctionnement de XMI.

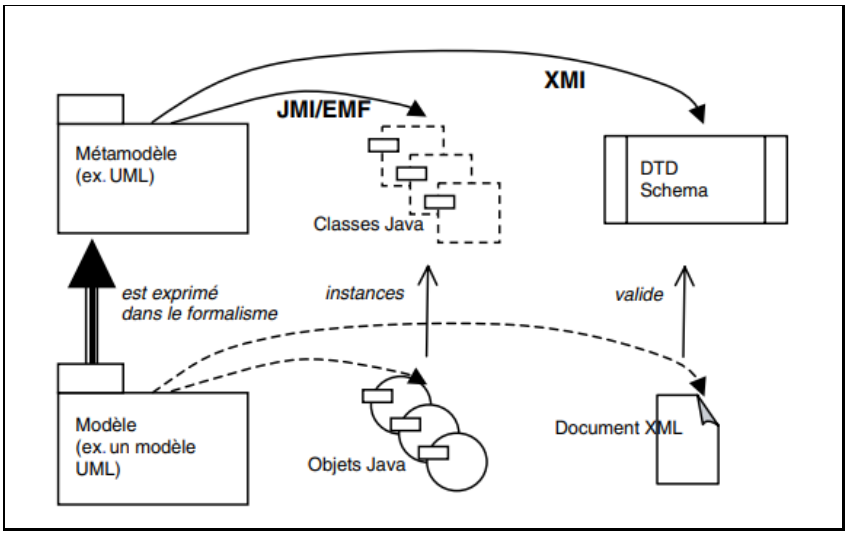


Figure 5.4 – Principe de fonctionnement de XMI selon [92]

5.3.2.3 QVT

Les transformations de modèles jouent un rôle central dans l'approche MDA. Afin d'accomplir ces transformations de type Model-To-Model (M2M), il est essentiel d'avoir à disposition un langage adapté. Pour répondre à cette nécessité, l'OMG a établi la norme QVT (Query, View, Transformation). Ce langage de haut niveau est intégré dans l'environnement EMF, permettant ainsi d'exprimer les transformations de modèles de manière efficace. Son

utilisation facilite le passage automatique entre les différentes phases de l'approche MDA.

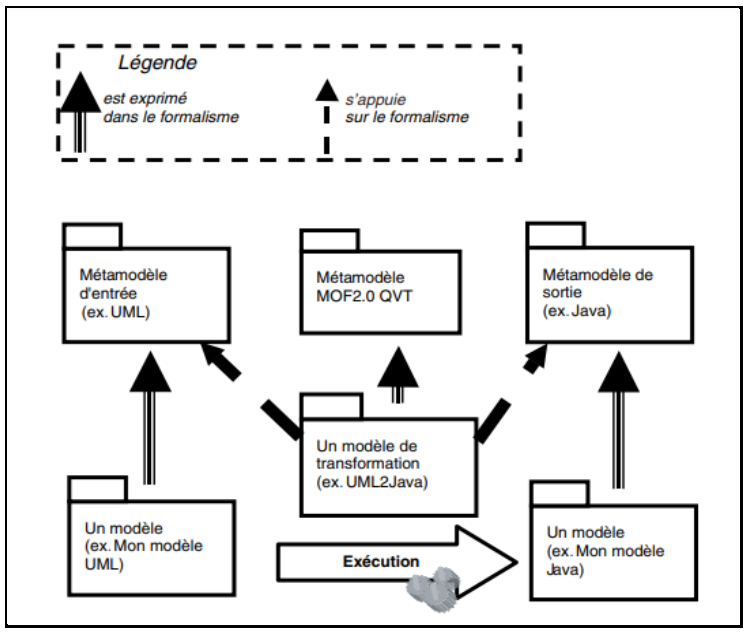


Figure 5.5 – Principe d'une transformation QVT selon [92]

Le principe d'une transformation QVT est illustré dans la figure 5.5. Il repose sur l'expression de règles de correspondances structurelles entre les métamodèles source et cible de la transformation. Un moteur de transformation interprète ces règles pour réaliser la transformation. Ainsi, une transformation QVT prend un modèle source en entrée et génère un modèle cible en sortie. Chaque modèle est conforme à un métamodèle.

Notre choix d'utiliser le langage QVT repose sur deux points essentiels : d'une part, il s'agit d'une norme établie par l'OMG et d'autre part, la plateforme EMF que nous avons utilisée dispose de ce formalisme, offrant ainsi une compatibilité optimale pour la réalisation des transformations de modèles.

Le module de transfert des données est détaillé dans la section suivante.

5.3.3. Transfert des données

Le processus est déclenché par l'acteur-métier qui sélectionne parmi l'ensemble des BD du LD. Le module *Transfer* est exécuté selon le type de la BD. Ce module a pour objectif d'automatiser le transfert des données du LD vers un entrepôt NoSQL. Il est composé de deux sous-modules qui s'appliquent en fonction du type de la BD source :

- *TransferRel* : qui assure la transformation des BD relationnelles dans le format de l'entrepôt.
- *TransferNoSQL* : qui réalise le transfert des BD NoSQL.

Dans notre application, l'implantation des modules selon les principes de MDA consiste à modéliser l'entrée et la sortie d'un traitement et à fournir une suite de règles de transformation appliquées aux modèles pour passer de l'entrée à la sortie. Afin de préserver une certaine généralité des traitements, les modèles d'entrée et de sortie sont représentés par des métamodèles. Pour les sous-modules, le passage d'un modèle à un autre se fait en utilisant des transformations de type M2M formalisées en QVT.

A la fin du traitement, l'entrepôt contient un ensemble de classes présentant les propriétés suivantes :

- le nombre de classes de l'entrepôt est égal à la somme du nombre de tables et de collections contenues dans le LD (toutes BD confondues)
- chaque classe prend le nom de la table ou la collection d'origine préfixé par le nom de la BD dans le LD
- chaque enregistrement d'une classe bénéficie d'un identifiant (Rid) attribué par le système gérant l'entrepôt (OrientDB)
- les liens contenus dans les enregistrements de l'entrepôt ont été stockés sous leur format d'origine (clés étrangères ou liens *DBRef* de MongoDB).

Nous détaillons par la suite les deux sous-modules *TransferRel* et *TransferNoSQL* en précisant l'entrée, la sortie, les transformations et l'implantation.

5.3.3.1. Sous-module *TransferRel*

Ce sous-module est chargé de transformer une BD relationnelle en une BD NoSQL. Dans ce qui suit, nous détaillons le sous-module *TransferRel* en précisant l'entrée, la sortie, les transformations et l'implantation.

Entrée

L'entrée du sous-module *TransferRel* est une BD relationnelle conforme au métamodèle relationnel que nous avons proposé dans le chapitre 3 (cf. figure 3.7). Ce métamodèle montre les principaux éléments composant une BD relationnelle ainsi que leurs caractéristiques structurelles.

Sortie

La sortie du sous-module *TransferRel* est une BD NoSQL conforme au métamodèle NoSQL proposé dans le chapitre 3 (cf. figure 3.8).

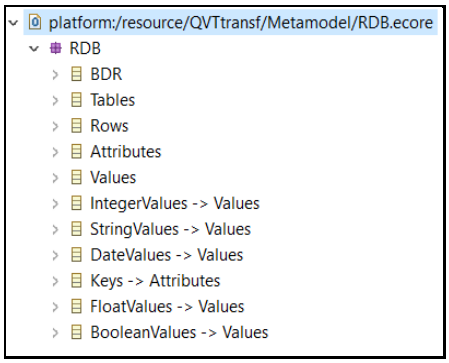
Transformation

Le sous-module *TransferRel* applique une transformation qui traduit une BD relationnelle en un modèle NoSQL conforme au métamodèle NoSQL que nous avons proposé dans le chapitre 3 (cf. figure 3.8). Cette transformation est réalisée par un ensemble de règles M2M formalisées en QVT.

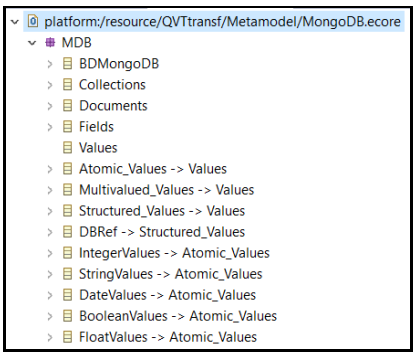
Implantation

Cette implantation nécessite la définition préalable d'un ensemble de métamodèles et de règles de transformation de type M2M.

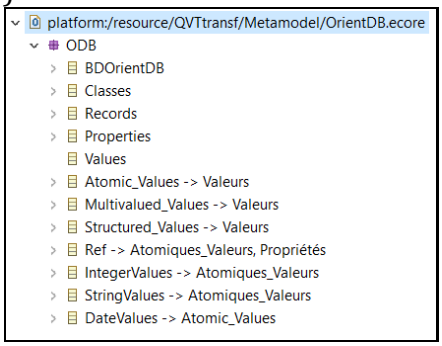
Nous avons utilisé l'environnement de développement EMF disposant du langage de métamodélisation Ecore. Tout d'abord, nous avons créé les métamodèles ECORE (relationnel et NoSQL) illustrés par la figure 5.6. Ces métamodèles décrivent respectivement la structure d'une BD relationnelle, MongoDB et OrientDB. Une description détaillée de ces métamodèles est donnée dans le chapitre 3 (cf. figures 3.7 et 3.8). Ecore se base sur XMI pour instancier les modèles.



(a)



(b)



(c)

Figure 5.6 – Métamodèles Ecore (a.relationnel, b.MongoDB et c.OrientDB) du module *Transfer*

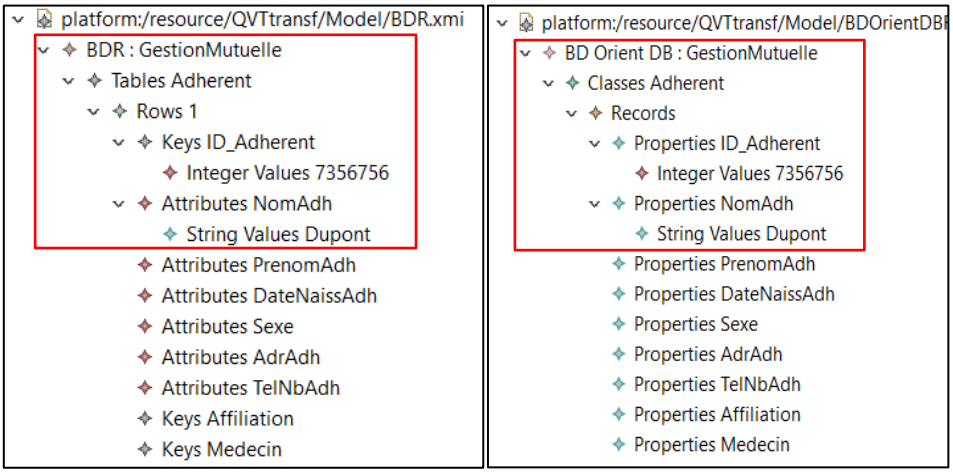
Ensuite, nous avons utilisé le langage QVT pour implanter les règles de transformation assurant le passage d'une BD relationnelle en une BD NoSQL. La figure 5.8 montre un extrait du script QVT correspondant ; les commentaires figurant dans les scripts indiquent les règles utilisées.

Finalement, nous avons testé le sous-module *TransferRel* (valider les règles QVT) en suivant les étapes suivantes :

- **Étape 1 :** Instancier le métamodèle du PIM relationnel pour produire la BD relationnelle (cf. figure 5.5.a) de l'application médicale décrite dans le chapitre 1 (cf. figure 5.7.a).
- **Étape 2 :** Exécuter le script QVT de *TransferRel* sur le PIM de la BD relationnelle d'entrée. Le résultat est illustré dans la figure 5.7.b. Il s'agit du PIM de la BD OrientDB (l'entrepôt).

Ces étapes ont permis de valider les règles QVT élaborées. Nous avons automatisé le transfert des données des BD sources vers la BD cible grâce à un script JAVA. Ce script permet de créer les classes relatives à chaque table (structures) dans l'entrepôt. Les données sont ensuite extraites du fichier contenant l'extension de la BD relationnelle source et chargées dans l'entrepôt.

La figure 5.12 montre les différentes étapes d'implantation du sous-module *TransferRel*.



(a) (b)

Figure 5.7 – PIM (a. BD relationnelle et b. BD OrientDB) du sous-module *TransferRel*

La figure 5.8 montre un extrait du script QVT permettant de transformer une BD relationnelle en une DB OrientDB (entrepôt).

```
//Transform a relational database into an OrientDB database
mapping BDR::RDBtoODB(): BDOrientDB{
dbName:=self.DBName;
classes:= self.tables.map toTable();
}

//Transform a relational table into an OrientDB class
mapping RDB::Tables:: toTable():ODB::Classes{
ClName := self.TableName;
records:= self.rows.map toDoc();
}

//Transform a row into an OrientDB Record
mapping RDB::Rows:: toDoc():ODB::Records{
couples:=self.hasAttributes.map toCouple();
```

Figure 5.8 – Script QVT de la transformation *TransferRel*

5.3.3.2. Sous-module TransferNoSQL

Ce sous-module est chargé de transformer une BD NoSQL en une autre BD NoSQL de même modèle (orienté document). Cette étape est rendue nécessaire pour effectuer un recalcul des adresses mémoire et des liens. Dans ce qui suit, nous détaillons le sous-module *TransferNoSQL* en précisant l'entrée, la sortie, les transformations et l'implantation.

Entrée

L'entrée du sous-module *TransferNoSQL* est une BD conforme au métamodèle NoSQL que nous avons proposé dans le chapitre 3 (cf. figure 3.8). Ce métamodèle donne les principaux éléments composant une BD NoSQL ainsi que leurs caractéristiques structurelles.

Sortie

La sortie du sous-module *TransferNoSQL* est une BD NoSQL.

Transformation

Le sous-module *TransferNoSQL* applique un ensemble de transformation qui traduit une BD NoSQL en un autre modèle NoSQL conforme au métamodèle NoSQL que nous avons proposé dans le chapitre 3 (cf. figure 3.8). Cette transformation est réalisée par un ensemble de règles M2M formalisées en QVT.

Les concepts sous-jacents des BD NoSQL en matière de représentation des données sont identiques et conformes au métamodèle NoSQL de la figure 3.8, mais le vocabulaire est différent. Il convient de noter que OrientDB offre une capacité de description plus riche, notamment en matière d'expression des liens. Dans un souci de clarté de la présentation, nous utiliserons le vocabulaire de MongoDB pour la source et le vocabulaire d'OrientDB pour la cible (Tableau 5.1). Mais ceci ne préjuge pas de la source qui peut être une BD OrientDB ; ce cas pose d'ailleurs des problèmes spécifiques et a été étudié (cf. section 3.3.2).

Vocabulaire de la source (MongoDB)	**Vocabulaire de la cible (OrientDB)**
Collection	Classe
Document	Enregistrement
Champ	Propriété
DBRef	Référence
Oid (Object identifier)	RID (Record IDentifier)

Tableau 5.1 – Correspondance des vocabulaires de la source et de la cible

Implantation

Ce sous-module permet de générer une BD NoSQL à partir d'un métamodèle NoSQL. Pour ce faire, nous avons utilisé le langage QVT pour implanter les règles de transformation du sous-module *TransferNoSQL*. La figure 5.10 montre un extrait du script QVT correspondant à ces règles.

Nous avons testé le sous-module *TransferNoSQL* en suivant les étapes suivantes :

- **Étape 1 :** Instancier le métamodèle NoSQL pour produire la BD NoSQL (cf. figures 5.6.b ou 5.6.c) de l'application médicale décrite dans le chapitre 1. Cette BD est stockée sous la forme d'un fichier XMI.
- **Étape 2 :** Exécuter le script *TransferNoSQL* sur la BD NoSQL d'entrée. Le résultat est illustré dans la figure 5.6.c.

Ces étapes ont permis de valider les règles QVT élaborées. Nous avons automatisé le transfert des données des BD sources vers la BD cible. Un script JAVA a permis de reproduire la structure de la BD NoSQL source dans

l'entrepôt. Les données sont ensuite extraites de la source et chargées dans l'entrepôt.

La figure 5.12 montre les différentes étapes d'implantation du sous-module *TransferNoSQL*.

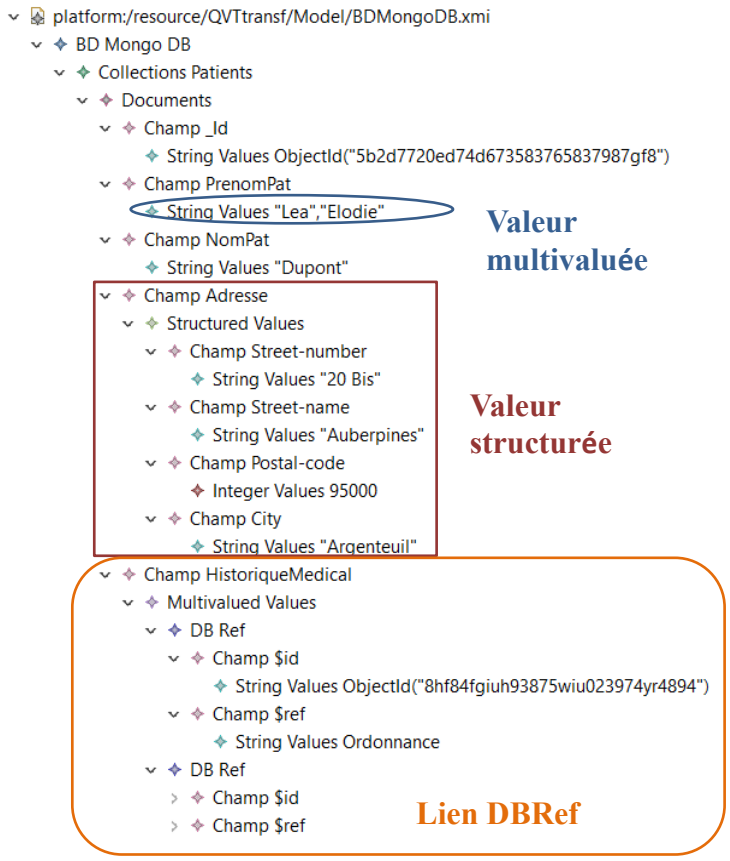


Figure 5.9 – PIM de la BD source (MongoDB) du sous-module *TransferNoSQL*

```
modeltype MDB "strict" uses 'http://mongodbModel.com';
modeltype ODB "strict" uses 'http://orientdbModel.com';

transformation NoSDDBToNoSDW(in source: MDB, out target: ODB );

main() {
source.rootObjects()[BDMongoDB] -> map MDBtoODB();
}
//Transform a MongoDB database into an OrientDB database
mapping BDMongoDB::MDBtoODB(): BDOrientDB{
dbName:=self.DBName;
classes:= self.collections.map toClass();
}
//Transform a MongoDB Collection into an OrientDB Class
mapping MDB::Collections:: toClass():ODB::Classes{
ClName := self.CollName;
records:= self.documents. map toDoc();
}
//Transform a MongoDB Document into an OrientDB Record
mapping MDB::Documents:: toDoc():ODB::Records{
couples := self.couples.map toAttr();
couples := self.couples.map ownAttributes();
}
```

Figure 5.10 – Script QVT de la transformation *TransferNoSQL*

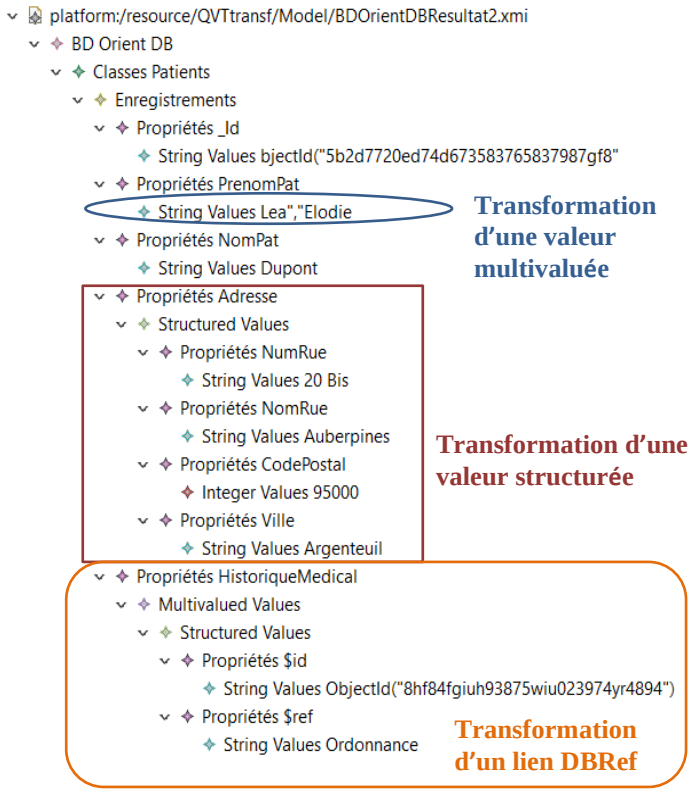


Figure 5.11 – PIM de la BD cible (OrientDB) du sous-module *TransferNoSQL*

Dans la figure 5.12, nous présentons les étapes du transfert des données relationnelles et NoSQL selon l'architecture MDA. Cette figure a été agrandie en Annexe A.

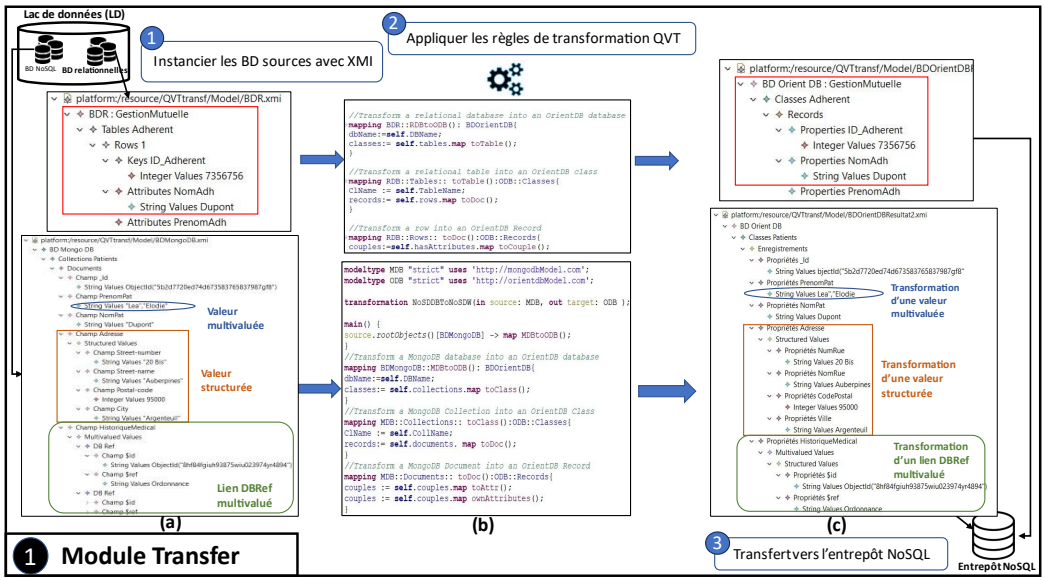


Figure 5.12 – Étapes du transfert des données relationnelles et MongoDB selon l'architecture MDA

La Figure 5.13 illustre le résultat du transfert des données des sources relationnelles et MongoDB vers la cible OrientDB. Cette figure a été agrandie en Annexe B.

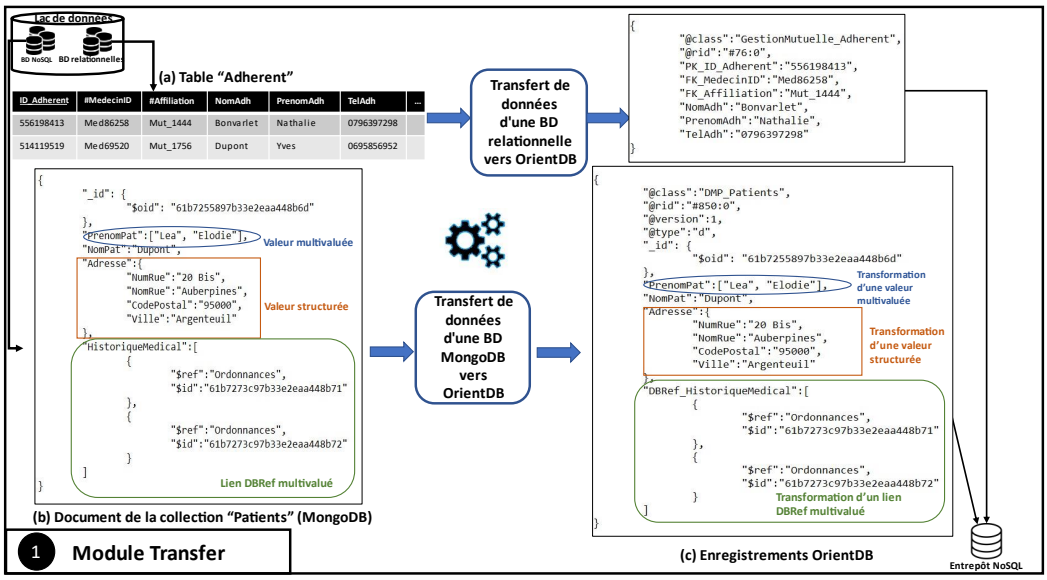


Figure 5.13 – Flux de données et traitements dans le module *Transfer*

Ainsi, on obtient une liste de classes correspondantes aux différentes tables, collections et classes transférées depuis le LD. Le résultat du transfert dans l'entrepôt OrientDB est représenté dans la figure 5.14 (cf. Annexe E). Dans l'exemple, la table « Adherent » de la BD « GestionMutuelle » est transformée en une classe « GestionMutuelle_Adherent ». La transformation des liens est détaillée dans la section 5.3.5.

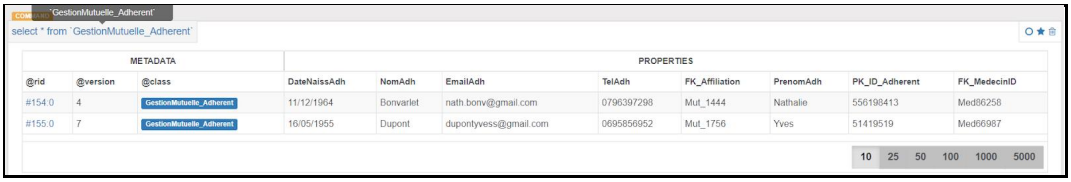


Figure 5.14 – Résultat du transfert après exécution du module *Transfer*

5.3.4. Fusion des données similaires

Dans un deuxième temps, le module *Merging* est appliqué à l'entrepôt et il a pour objectif de fusionner les classes considérées comme « similaires ». Pour réaliser la fusion, ce module utilise une ontologie[24] de domaine « Merge » fournie par des experts-métiers de notre cas d'étude. Cette ontologie, dont un extrait est présenté dans le tableau 5.2, a été stockée dans une BD relationnelle MySQL.

Par exemple, les classes « Patients », « Adherents » et « Assures » font partie du groupe d'équivalence « Patients ». Chacune de ces classes possède des propriétés comme « NomPat », « PrenomPat » et « TelPat » pour la classe "Patient" et « Nom_Adherent », « Prenom_Adherent » et « Telephone_Adherent » pour la classe « Adherents ». Cependant, certaines de ces propriétés sont équivalentes comme « NomPat » et « Nom_Adherent ». Ces deux propriétés sont stockées sous le nom « NomPatient » dans la nouvelle classe de l'entrepôt.

Groupe d'équivalence	Classes du groupe	Propriétés des classes	Propriété équivalente
Patients_DW	Patients Adherents Assures ….	NomPat NomAss NomAssure Nom_Adherent …	NomPatient
		PrenomPat PrenomAss PrenomAssure	PrenomPatient

[24] Les principes d'exploitation de l'ontologie ne sont pas détaillés dans cette thèse

		Prenom_Adherent …	
		TelPat TelAss TelAssure Telephone_Adherent …	TelPatient
		NumeroSecu NumS ID_Patient …	NSS
Medecins_DW	Docteurs Medecins …	NomMed NomDoct …	NomMedecin
		PrenomMed PrenomDoct …	PrenomMedecin
		TelMed TelDoct …	TelMedecin

Tableau 5.2 – Extrait de l'ontologie « Merge »

La figure 5.15 montre la fusion de deux enregistrements dans l'entrepôt. Dans l'exemple, l'ontologie indique que les classes « GestionMutuelle_Adherents » et « DMP_Patients » appartiennent à un même groupe d'équivalence. Ces deux classes sont issues de deux BD distinctes (« GestionMutuelle » et « DMP ») dans le LD. L'ontologie fournit toutes les propriétés de la classe unique « Patients_DW » élaborée par le module *Merging* et précise la classe d'où seront extraites ces propriétés. Dans le cas où des propriétés « similaires » (par exemple le « NomAdh » et « NomPat ») se trouvent dans plusieurs classes, l'ontologie indique la classe qui sera choisie pour extraire les valeurs et les enregistrer dans « Patients_DW ». Cette dernière est issue de la fusion de plusieurs classes ; son nom, qui figure dans l'ontologie, a été attribué par les experts. Cette figure a été agrandie en Annexe C.

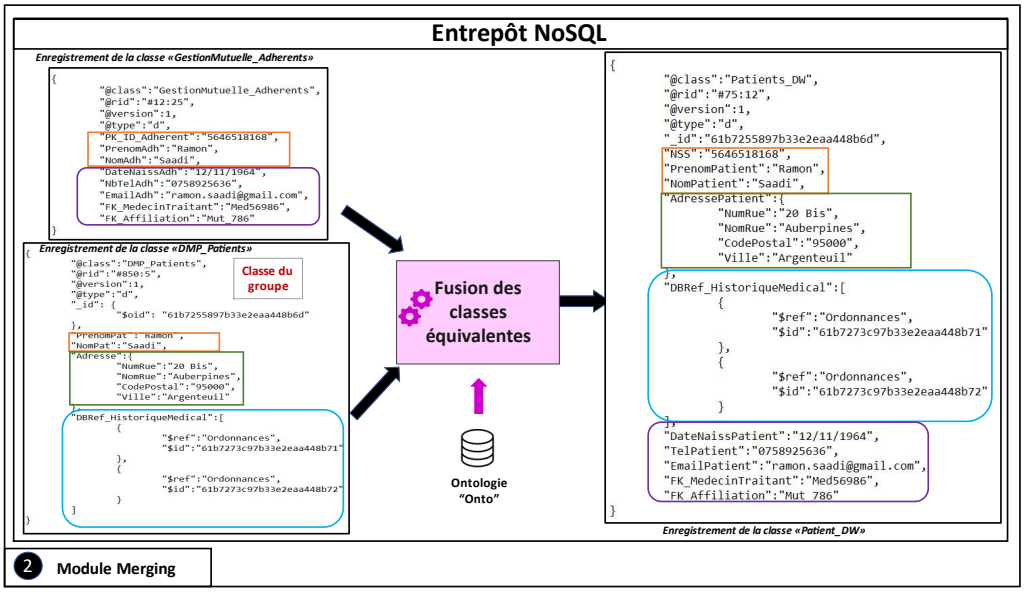


Figure 5.15 – Flux de données et traitements dans le module *Merging*

A ce niveau, les étiquettes (« FK_ » et « DBRef_ ») sont conservées. Elles permettent au module *Convert* de convertir les valeurs des propriétés en références. Le résultat de cette fusion dans l'entrepôt est illustré dans la figure 5.16 (cf. Annexe F).

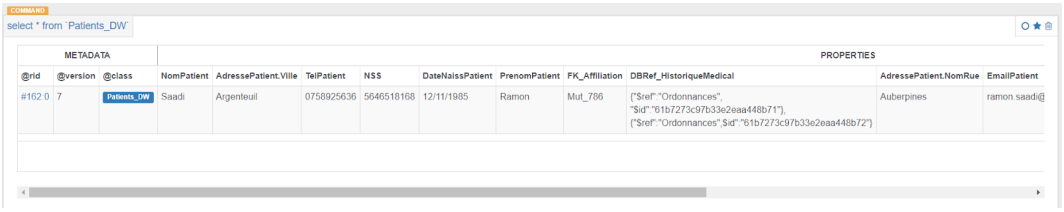


Figure 5.16 – Résultat de la fusion après exécution du module *Merging*

5.3.5. Conversion des liens

Le module *Transfer* a enregistré dans l'entrepôt l'ensemble des liens dans leurs formats d'origine (clé étrangère ou lien *DBRef*). Le module *Convert* va transformer chacun de ces liens en références Rid ; ces liens seront ainsi conformes aux principes objets définis par l'ODMG. Il comporte deux traitements : *ConvertRel* et *ConvertNoSQL* et nécessite un balayage complet de l'entrepôt. Dans un enregistrement, les propriétés-liens ont été étiquetées par le module *Transfer*. Si une telle propriété contient une clé étrangère, le traitement *ConvertRel* s'applique ; si elle contient un lien *DBRef* alors le traitement *ConvertNoSQL* est activé. Dans les deux cas, le module *Convert* substitue les valeurs des propriétés en références (Rid des enregistrements initiaux) comme le montre l'exemple de la figure 5.17.

Dans cet exemple, la partie gauche de la figure représente quatre enregistrements de l'entrepôt issus des classes « Ordonnances_DW », « Mutuelles_DW » et « Medecins_DW » avant la conversion des liens et la partie droite, le résultat dans la classe « Patients_DW » après la conversion.

Dans la partie gauche, l'enregistrement initial de la classe « Patients_DW » contient deux clés étrangères (« FK_MedecinTraitant » et « FK_Affiliation ») issues de deux tables relationnelles. Grâce à un dictionnaire (métadonnées) associant les clés primaires aux identifiants Rid d'orientDB, le module *Convert* substitue un Rid à chaque clé étrangère. Un traitement identique est appliqué au second enregistrement contenant un lien *DBRef* (partie gauche) ; il s'agit d'un lien multivalué MongoDB (« DBRef_HistoriqueMedical ») ; dans ce cas, le dictionnaire utilisé associe les Oid de MongoDB aux Rid d'OrientDB. Dans la partie droite de la figure 5.17, l'enregistrement de la classe « Patients_DW » a été mis à jour. Cette figure a été agrandie en Annexe D.

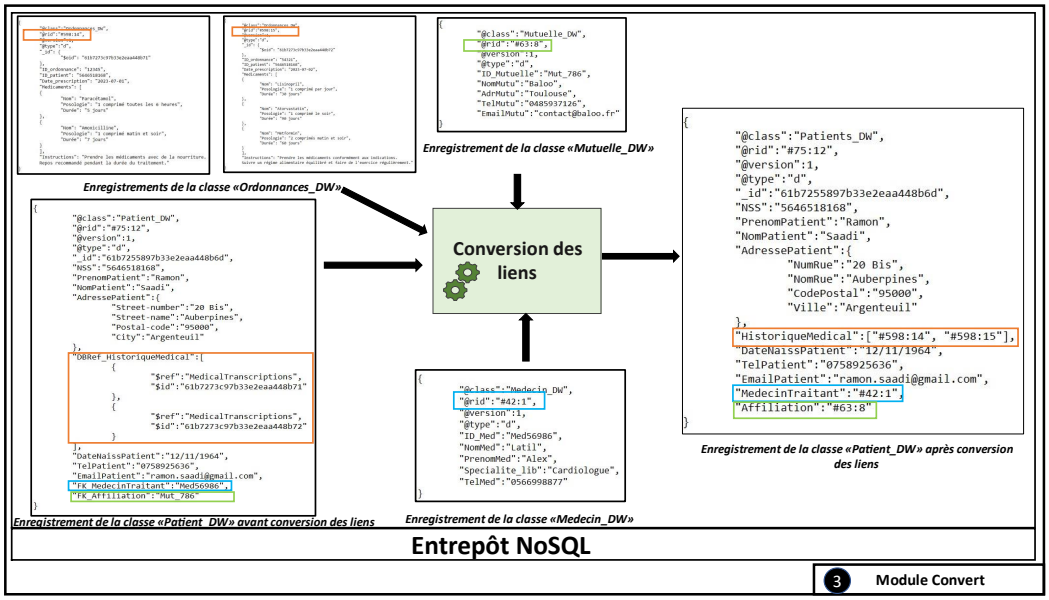


Figure 5.17 – Conversion des liens relationnels et DBRef dans le module *Convert*

Le résultat de la conversion de liens dans l'entrepôt OrientDB est illustré dans la figure 5.18 (cf. Annexe G).

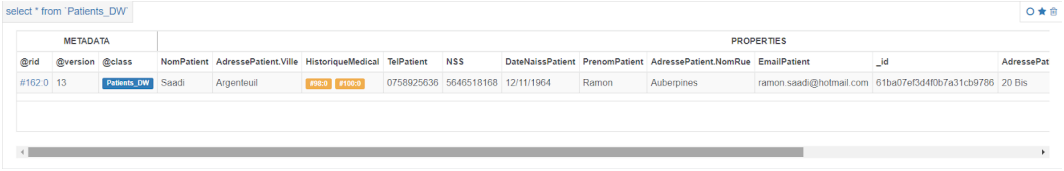


Figure 5.18 – Résultat de conversion des liens après exécution du module *Convert*

5.4. Processus DWtoDM

L'un des verrous de notre thèse concerne l'accès aux données de l'entrepôt par des acteurs-métier. Dans notre environnement applicatif, l'utilisateur-type de notre solution est un non informaticien qui maîtrise bien son besoin-métier mais qui est peu rompu à l'élaboration de requêtes d'accès complexes. L'objectif est d'apporter une assistance à des acteurs-métier par des mécanismes de personnalisation et de recommandation pour les aider à obtenir un schéma de magasin correspondant le mieux à leurs besoins et aux entités qu'ils souhaitent manipuler. Pour chaque acteur-métier, le système permet soit de personnaliser le schéma d'un magasin soit de recommander une liste de schémas ; il renvoie le résultat à l'acteur-métier qui peut modifier le schéma proposé.

5.4.1. Architecture technique de DWtoDM

Le prototype DWtoDM repose principalement sur un entrepôt implanté dans le SGBD OrientDB. Le système est géré par une application cliente JAVA qui permet la personnalisation et la recommandation de schémas de magasin auprès de l'acteur-métier. La Figure 5.19 illustre l'architecture de notre système.

- Le niveau « Stockage » représente l'implantation du schéma et des données de l'entrepôt.
- Le niveau « Traitements » comporte d'une part un mécanisme de personnalisation et permettant de créer un schéma de magasin et d'autre part un mécanisme de recommandations qui propose à l'acteur-métier une liste de schémas de magasin recommandée.
- Le niveau « Interactions » offre un ensemble d'interfaces permettant à l'acteur-métier d'interagir avec le système.

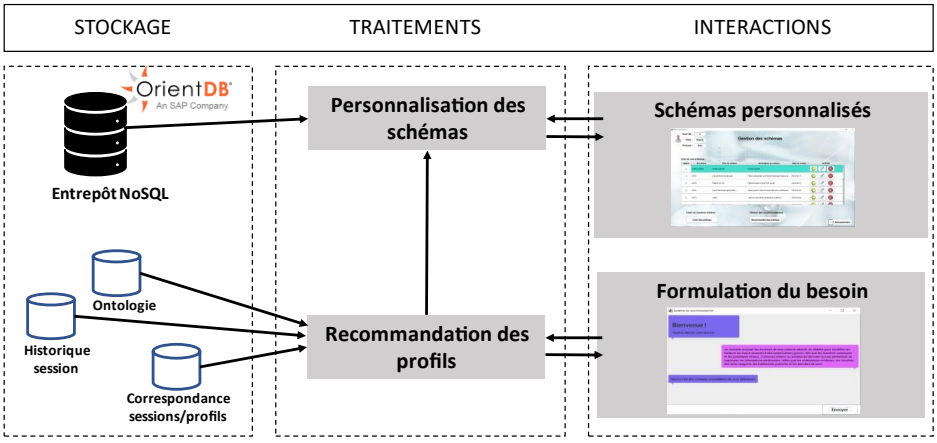


Figure 5.19 – Architecture technique de DWtoDM

La Figure 5.20 présente un diagramme d'états-transitions UML qui décrit les étapes d'évolution du processus de construction d'un schéma en fonction des interactions de l'acteur-métier. Le processus DWtoDM intervient à différents niveaux et permet de :

- aider l'acteur-métier à construire un schéma à partir du schéma global ou d'un schéma existant (personnalisation).
- recommander des schémas élaborés par d'autres acteur-métiers partageant les mêmes intérêts.
- assister l'acteur-métier dans la personnalisation du schéma recommandé une fois validé, puis charger les données dans le magasin afin de permettre son interrogation.

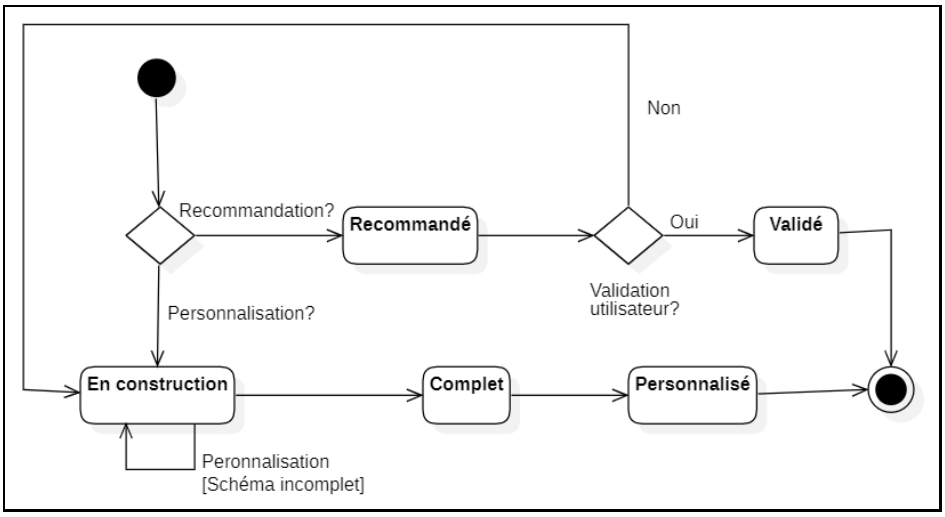


Figure 5.20 – Evolution de l'élaboration d'un schéma de magasin

Dans ce qui suit, nous détaillerons le processus de personnalisation des schémas de magasins.

5.4.2. Personnalisation

Nous avons développé une interface simple permettant aux utilisateurs d'interagir avec notre système.

Le point de départ consiste à se connecter au système via une interface d'authentification. Ensuite, l'utilisateur se verra proposer la liste des schémas créés par lui-même lors de consultations antérieures. Il a donc la possibilité d'exploiter un schéma préalablement créé ou bien d'en créer un nouveau.

Figure 5.21 – Interface proposant l'historique des schémas d'un utilisateur

Dans le processus DWtoDM, l'utilisateur a la possibilité d'ajuster la portée du magasin en fonction de ses besoins spécifiques. Il peut choisir d'augmenter ou de restreindre les données incluses dans le magasin.

5.4.2.1. Processus d'augmentation d'un magasin de données

Pour élaborer un nouveau schéma, l'utilisateur s'appuie sur le schéma global ou le schéma d'un magasin existant. Il sélectionne une ou plusieurs classes correspondant à son besoin. Nous montrons ci-après un exemple d'augmentation d'un magasin.

Supposons que l'utilisateur souhaite créer un magasin pour obtenir des performances opérationnelles dans un service spécifique des hôpitaux, tel que le service des urgences. Il peut choisir d'inclure uniquement les données relatives aux patients, aux admissions, aux temps d'attente, aux diagnostics et aux traitements liés aux urgences. L'utilisateur choisit alors les classes qui l'intéressent. Il a aussi la possibilité de sélectionner des classes liées à l'une de ces classes précisant un type de lien ; puis il sélectionne les attributs qui l'intéressent. La figure 5.22 montre une capture de l'interface permettant la création d'un nouveau schéma à partir du schéma global de l'entrepôt.

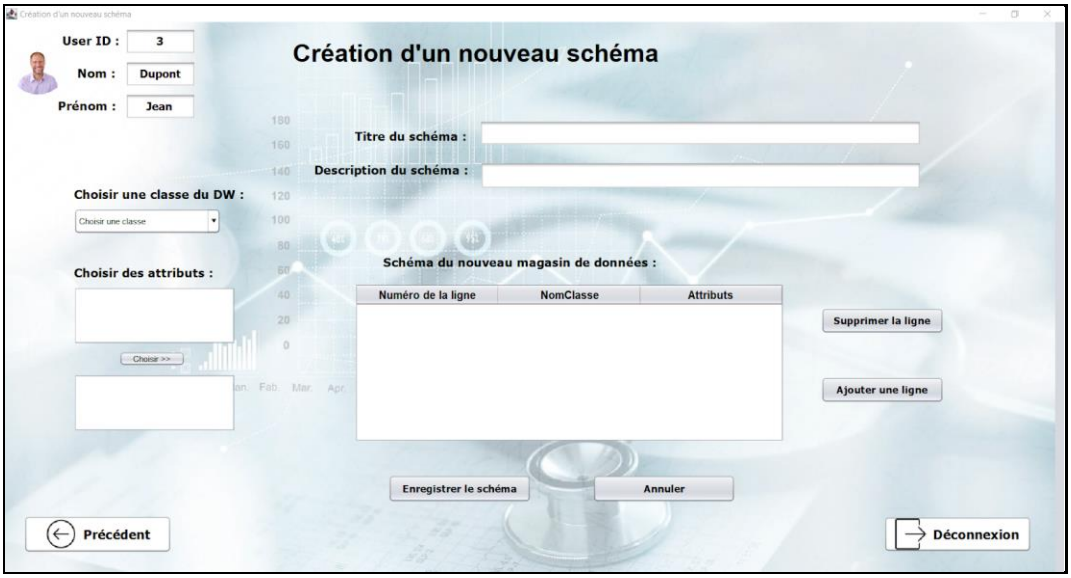


Figure 5.22 – Interface augmentation d'un magasin

Pour créer un nouveau schéma de magasin à partir du schéma global :

1. L'utilisateur accède à l'interface de création de magasin à partir du schéma global.

2. Il choisit les classes principales qui sont directement liées aux performances opérationnelles du service des urgences, telles que :

   - Classe « Patients_DW » : pour suivre les informations démographiques et médicales des patients traités aux urgences.
   - Classe « Admissions_DW » : pour enregistrer les détails d'admission des patients aux urgences, tels que la date et l'heure d'admission, le mode d'admission, etc.
   - Classe « Diagnostics_DW » : pour enregistrer les diagnostics initiaux des patients effectués aux urgences.
   - Classe « Traitements_DW » : pour suivre les traitements initiaux administrés aux patients aux urgences.

   (Rappel : les acteurs-métier sont familiers avec la manipulation des entités contenues dans ces classes)

3. Ensuite, il peut choisir d'autres classes pouvant être liées aux classes sélectionnées. Par exemple :

   - La classe « Patients_DW » est liée à la classe « AntecedentsMedicaux_DW » pour relier les informations relatives aux antécédents médicaux d'un patient spécifique.

- La classe « Traitements_DW » est liée à la classe « Medicaments_DW » pour relier les informations sur les médicaments administrés dans le cadre d'un traitement spécifique.

4. Enfin, il sélectionne les attributs qui l'intéressent pour chaque classe. Par exemple :
   - Pour la classe « Patient_DW » : NSS, nomPatient, prenomPatient, dateNaissance, sexe, etc.
   - Pour la classe « Admissions_DW » : date_admission, heure_admission, motif_admission.
   - Pour la classe « Diagnostics_DW » : code_diagnostic, description_diagnostic.
   - Pour la classe « Traitements_DW » : medicaments_administres, procedures_effectuees.
   - Pour la classe « AntecedentsMedicaux_DW » : maladies_passees, conditions_medicales_preexistantes, interventions_chirurgicales_anterieures, medicaments_pris_regulierement, allergies.
   - Pour la classe « Medicaments_DW » : nomMed, nomMed, descriptionMed, dosage, posologie, date_prescription, date_debut, date_fin.

5.4.2.2. Processus de restriction d'un magasin de données

Il est également possible pour l'utilisateur de restreindre le magasin en filtrant les données selon des critères spécifiques. Ainsi, pour chaque session effectuée par un utilisateur, le fichier « QLog » contient l'ensemble des requêtes qui ont été appliquées sur le magasin. Considérons un utilisateur ayant réalisé une session avec magasin pour étudier les traitements contre le cancer du sein. Il utilise un magasin contenant des données cliniques sur différents types de cancer, notamment les diagnostics, les traitements et les résultats des patients.

Supposons que le magasin initial contient des classes telles que « Patients_DW », « Diagnostics_DW », « Traitements_DW », « Medicaments_DW », « AntecedentsMedicaux_DW » et "Resultats_DW » et des attributs tels que « NSS », « type_diagnostic », « type_traitement », « duree_traitement », « resultat_traitement », etc.

Les acteurs-métier posent des requêtes pour extraire les données nécessaires à leurs besoins. Par exemple, ils peuvent utiliser des requêtes telles que :

- « Sélectionner les patients atteints de cancer du sein avec des résultats de traitement positifs »
- « Afficher les patients atteints de cancer du sein diagnostiqués au stade avancé »
- « Quels sont les traitements de chimiothérapie administrés aux patients atteints de cancer du sein »
- « Je souhaite avoir la liste des patients atteints de cancer du sein avec des résultats de traitement négatifs »
- « Afficher les patients atteints de cancer du sein qui ont reçu un traitement de radiothérapie pendant plus de 6 semaines »
- « Liste des patients atteints de cancer du sein avec un certain facteur de risque (par exemple, antécédents familiaux de cancer) »

Ces requêtes sont enregistrées dans un fichier de log (QLog) pour être utilisées ultérieurement comme décrit dans la section 4.3.1. Chacune d'elles étant exprimée sous forme SQL, on conserve les clauses SELECT et WHERE.

À la fin de la session, le système propose aux chercheurs de restreindre le magasin en fonction des données manipulées. Le module de restriction parcourt la BD « QLog » et identifie les classes qui ont été utilisées dans les requêtes. Dans notre exemple, il déduit que les classes « Patients_DW », « Diagnostics_DW », « Traitements_DW » et « Resultats_DW » ont été manipulées. Le magasin est restreint pour ne contenir que les classes utilisées par les requêtes. Les acteurs-métier peuvent maintenant continuer à interroger le magasin restreint qui contient uniquement les données spécifiques à leurs besoins. Ils peuvent effectuer des analyses statistiques ou des comparaisons de résultats de traitement spécifiques au cancer du sein. En restreignant le schéma du magasin, le système permet à l'acteur-métier de se concentrer sur les informations pertinentes. Ce magasin restreint peut par la suite être étendu comme détaillé dans la section 5.4.2.1 précédente.

A ce stade, aucune recommandation de schéma n'est effectuée. L'utilisateur peut poser ses requêtes sur le schéma du magasin. Chaque requête est stockée dans le fichier « QLog ». Cette étape permettra, par la suite, au système d'apprendre et d'affiner les recommandations à l'utilisateur lors de ces prochaines connexions. Un extrait de cette BD est donné dans la figure 5.24.

UserID	SessionID	QueryID	QueryPart	QueryText	DateHeure	MotsUtiles	ClassesAssociées	AttributsAssociés
1001	1	1	1	Trouver les patients diabétiques	2023-08-20 09:30:00	patients, diabétiques	Patients	ÉtatDeSanté
1001	2	1	1	Symptômes des rhumes	2023-08-19 10:00:00	Rhume, Symptômes	Maladies	Symptômes, Traitement
1001	3	1	1	Trouver patients atteints cancer	2023-08-20 14:30:00	patients, cancer	Médecine, Oncologie	Diagnostic, Traitements, Dossiers médicaux
1001	4	1	1	Symptômes grippe et traitements	2023-08-20 15:45:00	symptômes, grippe, traitements	Médecine, Infectiologie	Symptômes, Médicaments, Prévention
1002	1	1	1	Prise de tension	2023-08-19 12:45:00	Tension, Mesure	Hypertension	Méthodes de mesure
1002	2	1	1	Types de cancer	2023-08-20 09:15:00	Cancer, Types	Oncologie	Caractéristiques
1002	3	1	1	Diagnostics maladie cardiaque	2023-08-21 10:20:00	diagnostics, maladie cardiaque	Médecine, Cardiologie	Tests, Échocardiogrammes, Traitements
1003	1	1	1	Chirurgie cardiaque	2023-08-21 14:30:00	Chirurgie, Coeur	Chirurgie	Techniques
1003	2	1	1	Afficher les dossiers de cardiologie	2023-08-20 10:15:00	dossiers, cardiologie	DossiersMédicaux	DomaineSpécialité
1003	3	1	1	Vaccins COVID-19	2023-08-19 11:30:00	Vaccin, COVID-19	Vaccinations	Effets secondaires
1003	4	1	1	Témoignages patients COVID-19	2023-08-21 14:00:00	témoignages, patients, COVID-19	Médecine, Infectiologie	Expériences, Symptômes, Précautions
1004	1	1	1	Rechercher les cas d'AVC des 6 derniers mois	2023-08-20 11:45:00	cas, AVC, 6 derniers mois	CasMédicaux	Diagnostic, Date
1004	2	1	1	Obtenir les antécédents médicaux de M. Smith	2023-08-20 14:20:00	antécédents, médicaux, M. Smith	AntécédentsMédicaux	PatientID
1004	3	1	1	Identifier les allergies alimentaires	2023-08-21 09:00:00	allergies, alimentaires	Allergies	TypeAllergie

Figure 5.24 – Extrait du fichier « QLog »

Le mécanisme de personnalisation a permis à chaque utilisateur de construire son propre schéma de magasin et en l'adaptant à ses besoins spécifiques. Il peut également aller plus loin en utilisant ces schémas pour proposer des recommandations susceptibles d'intéresser l'utilisateur. A présent, nous allons étudier un processus d'assistance basé sur des interactions entre le système et les utilisateurs.

5.4.3. Recommandation

Dans le cas où la création d'un magasin ne lui convient pas, L'utilisateur peut demander des recommandations de schémas. Ainsi, notre système propose une liste de schémas susceptibles d'intéresser l'utilisateur en fonction d'un besoin exprimé sous la forme d'une phrase en langage naturel ou, plus simplement, d'une suite de mots-clés. Par exemple, supposons qu'un médecin généraliste, pose une question via l'interface de dialogue du système (cf. figure 5.25) :

"Je souhaite avoir la liste des patients atteints de diabète pour identifier les facteurs de risque associés à des complications graves, telles que les maladies cardiaques et les problèmes rénaux. J'aimerais obtenir un schéma de données qui me permettrait de regrouper les informations pertinentes, telles que les antécédents médicaux, les résultats des tests sanguins, les traitements prescrits et les données de suivi."

Il est à noter que plus le médecin sera précis dans l'expression de la requête, plus le système sera efficace dans ses recommandations.

Dans cet exemple, le médecin exprime son besoin d'obtenir les données de ses patients diabétiques pour découvrir les facteurs de risque liés à des complications graves.

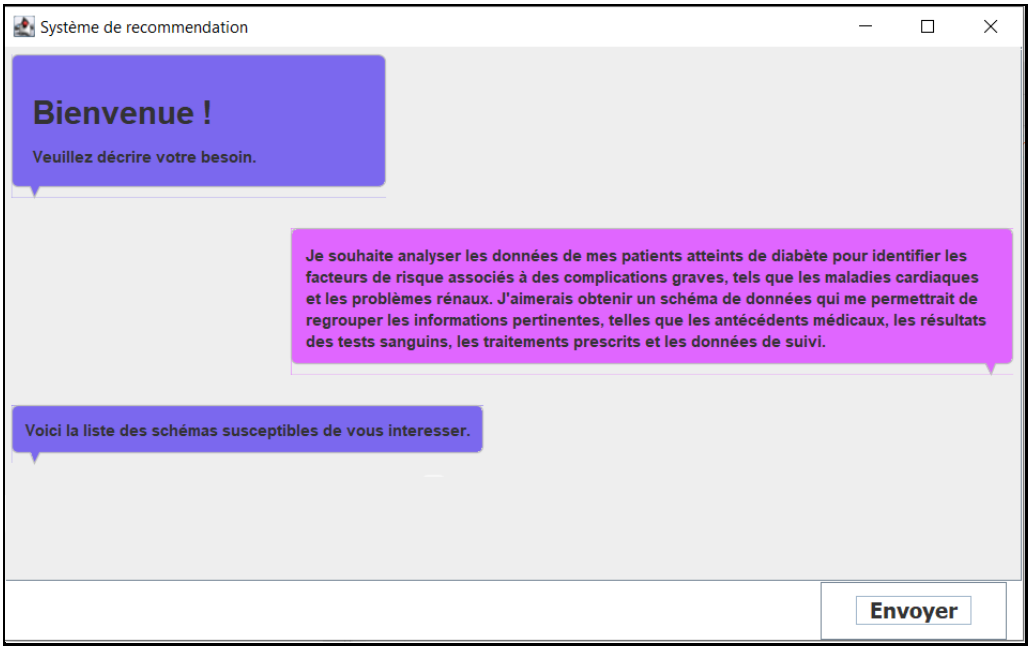


Figure 5.25 – Interface d'expression du besoin

Étape 1 : Élimination des terminaisons, des mots-outils et des signes de ponctuation :

Cette étape est basée sur des techniques largement reconnues dans le domaine du traitement du langage naturel (par exemple, la tokenisation et la lemmatisation) pour identifier les éléments clés du texte [93], [94]. Les terminaisons sont retirées, tandis que les mots-outils, tels que les prépositions et les conjonctions, sont filtrés pour ne conserver que les termes pertinents. De plus, les signes de ponctuation sont éliminés pour se concentrer sur le contenu sémantique.

En appliquant ces techniques, le système est en mesure d'extraire les mots utiles et les éléments significatifs du texte. Pour notre cas :

liste données patients atteints diabète identifier facteurs risque associés complications graves maladies cardiaques problèmes rénaux schéma données permettra regrouper informations pertinentes antécédents médicaux résultats tests sanguins traitements prescrits données suivi

Ces mots seront augmentés à l'aide de l'ontologie des synonymes de la figure 5.26.

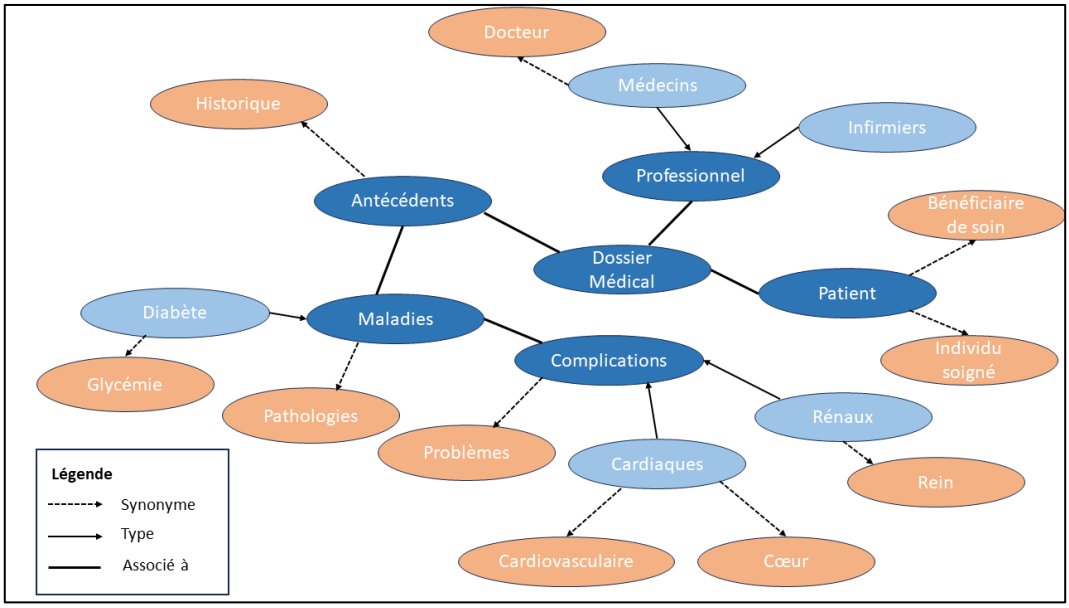


Figure 5.26 – Extrait de l'ontologie des synonymes

Étape 2 : Association des mots avec des attributs de l'entrepôt :

Le module Mappeur (cf. figure 4.7) associera chaque mot à des synonymes puis à des attributs de la BD médicale (entrepôt) en utilisant l'ontologie de domaine et la base de correspondance entre les mots et les attributs de la BD. Voici l'association des mots avec des attributs de l'entrepôt pour l'exemple donné (cf. Tableau 5.3) :

Mots	Attributs associés	Classe source
"patients"	Patients_DW.NomPatient	Patients_DW
"diabète"	DossierMedical_DW.SuiviDiabete	DossierMedical_DW
"complications graves"	DossierMedical_DW.Complications	DossierMedical_DW
"maladies cardiaques"	DossierMedical_DW.Complications	DossierMedical_DW
"problèmes rénaux"	DossierMedical_DW.Complications	DossierMedical_DW
"antécédents médicaux"	DossierMedical_DW.AntecedentsMedicaux	DossierMedical_DW
"résultats tests sanguins"	Tests_DW.Resultats	Tests_DW
"traitements prescrits"	Diagnostics_DW.TraitementsPrescrits	Diagnostics_DW
"données suivi"	DossierMedical_DW.HistoriqueConstantes	DossierMedical_DW

Tableau 5.3 – Association des mots avec des attributs de l'entrepôt

Le Mappeur associe ainsi l'ensemble des mots pertinents avec les attributs correspondants de la BD. Ce processus permet de convertir le besoin de l'utilisateur en un ensemble d'attributs de l'entrepôt, présenté sous forme de vecteur VB (cf. section 4.3.2.1) qui servira de base pour la recommandation de schémas de magasin adaptés à ce besoin.

Étape 3 : Calcul du poids des attributs :

Dans cet exemple, le vecteur VB est composé de 7 paires (A, w), où A représente un attribut de la BD déduit du besoin de l'utilisateur, et w représente son poids, c'est-à-dire le nombre de fois où cet attribut apparaît dans le besoin exprimé.

Le résultat final est le vecteur VB suivant :

VB = {(Patients_DW.NomPatient, 1), (DossierMedical_DW.Suivi_diabete, 1), (DossierMedical_DW.Complications, 3), (DossierMedical_DW.AntecedentsMedicaux, 1), (Tests_DW.Resultats, 1), (Consultation_DW.TraitementsPrescrits, 1), (DossierMedical_DW.HistoriqueConstantes, 1)}

Étape 4 : Proposition de schémas

Pour proposer des schémas à l'utilisateur, le processus utilise les 3 éléments suivants : le vecteur VB, le dictionnaire des attributs (DA) (cf. Tableau 5.4) et la matrice HS.

Pour notre cas d'étude, voici un extrait de la matrice HS :

HS =

Sessions \ Attributs	1	4	5	21	22	23	25	28	34	35	36	37	38	52	66	71
S1	1	1	1	0	0	3	1	2	1	0	3	1	1	0	2	0
S2	1	3	1	1	0	1	0	1	1	1	0	2	3	1	2	1
S3	1	0	1	3	1	1	2	1	2	0	1	4	1	0	2	0
S4	1	2	1	0	1	0	0	3	2	1	1	2	1	3	1	2
S5	1	1	0	1	0	0	1	1	0	0	0	0	2	5	4	3
S6	1	2	1	0	0	2	3	1	1	3	0	2	0	4	3	1
S7	1	0	0	0	0	3	1	4	1	0	3	1	1	0	0	0
S8	1	3	1	1	0	1	0	1	1	1	0	2	3	1	2	1
S9	1	1	0	1	0	0	1	1	0	0	0	0	2	3	1	1
S10	1	2	1	0	1	0	0	3	2	1	1	2	1	3	1	2

N° d'ordre	Nom de l'attribut	Type de l'attribut	Libellé de l'attribut	Classe source
1	Patients_DW.NomPatient	Chaîne de caractères	Nom du patient	Patients_DW
4	Patients_DW.Age	Entier	Age du patient	Patients_DW
5	Patients_DW.Sexe	Chaîne de caractères	Genre du patient	Patients_DW
21	DossierMedical_DW.HistoriqueConstantes	Chaîne de caractères	Constantes du patients	DossierMedical_DW
22	DossierMedical_DW Diabetique	Boolean	Si patient diabétique ou non	DossierMedical_DW
23	DossierMedical_DW.SuiviDiabete	Référence	Données de suivi du diabète	DossierMedical_DW
28	DossierMedical_DW.AntecedentsMedicaux	Chaîne de caractères	Liste des antécédents médicaux	DossierMedical_DW
34	Constantes_DW.IMC	Nombre réel	Indice de masse corporelle du patient	Constantes_DW
35	Constantes_DW.PressionArt	Nombre réel	Pression artérielle à un instant t	Constantes_DW
37	Constantes_DW.Glycemie	Nombre réel	Glycémie à un instant t	Constantes_DW
38	Constantes_DW.Date_suivi	Date/heure	Date de mesures	Constantes_DW
52	Tests_DW.Resultats	Chaîne de caractères	Résultats des tests sanguins	Tests_DW
66	Consultation_DW.TraitementsPrescrits	Référence	Liste des traitements prescrits	Consultation_DW
71	Traitements_DW.Medicaments	Référence	Liste des médicaments du patients	Traitements_DW

Tableau 5.4 – Extrait du dictionnaire de données DA

Ici, nous avons dix sessions passées représentées par les lignes de la matrice HS. Chaque session est associée à des attributs qui ont été manipulés au cours de cette session.

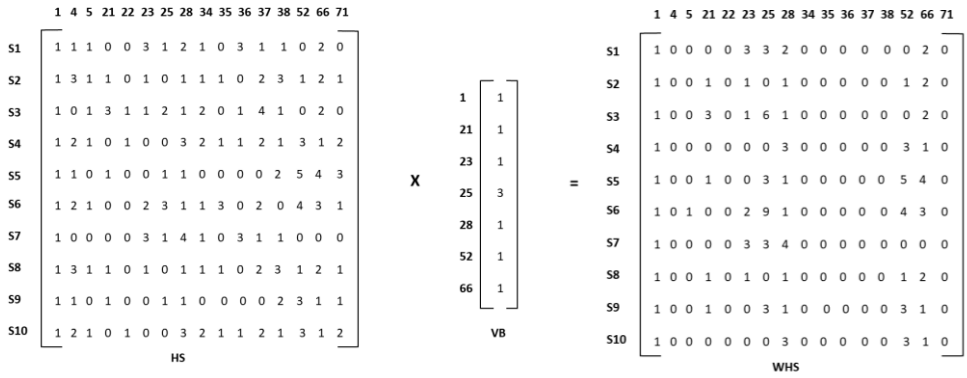
Par exemple, dans la Session 1, les attributs "Patients_DW.NomPatient", "DossierMedical_DW.Suivi_diabete", "DossierMedical_DW.Complications" et "DossierMedical_DW.AntecedentsMedicaux" étaient présents, tandis que les attributs "Tests_DW.Resultats" et "Consultation_DW.TraitementsPrescrits", "DossierMedical_DW.HistoriqueConstantes" étaient absents.

La matrice HS est utilisée par le module Prescripteur pour évaluer la similarité entre le besoin exprimé par l'utilisateur (représenté par le vecteur VB) et les sessions passées. Cette matrice permet de prédire les schémas de magasin les plus pertinents pour l'utilisateur.

Étape 5 : Pondération des attributs

Cette étape consiste à attribuer des poids aux attributs de la matrice HS.

Dans l'exemple suivant, on voit que pour la Session 3, l'attribut n° 25 « DossierMedical_DW.Complications » a un effectif de 2 et un poids de 3 dans VB ; son poids pondéré dans la session est de 2 * 3 = 6.



Cette pondération permet de marquer l'importance des attributs du besoin dans les différentes sessions passées, ce qui sera utilisé ultérieurement par le module Prescripteur pour recommander les schémas de magasin les plus adaptés à l'utilisateur.

Étape 6 : Restriction de la matrice WHS

Cette étape crée la sous-matrice RWHS en réduisant la matrice WHS aux seuls attributs de VB. Ceci permet de filtrer les attributs non pertinents. Dans notre exemple, la sous-matrice WHS sera réduite à la matrice suivante :

RWHS =

	1	21	23	25	28	52	66	71
S1	1	0	3	3	2	0	2	0
S2	1	1	1	0	1	1	2	0
S3	1	3	1	6	1	0	2	0
S4	1	0	0	0	3	3	1	0
S5	1	1	0	3	1	5	4	0
S6	1	0	2	9	1	4	3	0
S7	1	0	3	3	4	0	0	0
S8	1	1	1	0	1	1	2	0
S9	1	1	0	3	1	3	1	0
S10	1	0	0	0	3	3	1	0

Étape 7 : Pénalisation des attributs

Le système parcourt chaque attribut dans RWHS et affecte une pénalité de -1 pour chaque attribut non utilisé (poids = 0). Dans la matrice RWHS de l'étape 6, les attributs n° 21 et 52 de la session 1 n'ont pas été utilisés, ils ont initialement un poids égal à 0. Le système leur attribue alors un poids de -1. Après pénalisation, La matrice RWHS est transformé comme suit :

RWHS =

	1	21	23	25	28	52	66
S1	1	-1	3	3	2	-1	2
S2	1	1	1	-1	1	1	2
S3	1	3	1	6	1	-1	2
S4	1	-1	-1	-1	3	3	1
S5	1	1	-1	3	1	5	4
S6	1	-1	2	9	1	4	3
S7	1	-1	3	3	4	-1	-1
S8	1	1	1	-1	1	1	2
S9	1	1	-1	3	1	3	1
S10	1	-1	-1	-1	3	3	1

Étape 8 : Calcul du score

En cumulant les effectifs pondérés des attributs utilisés par une session (c'est-à-dire une ligne de RWHS), nous obtenons le score de la session au

regard du besoin VB de l'utilisateur. Ce score représente la similitude entre le besoin exprimé par l'utilisateur et une session passée.

S1 = 1 - 1 + 3 + 3 + 2 - 1 + 2 = 9

S2 = 1 + 1 + 1 - 1 + 1 + 1 + 2 = 6

S3 = 1 + 3 + 1 + 6 + 1 - 1 + 2 = 13

S4 = 1 - 1 - 1 - 1 + 3 + 3 + 1 = 5

S5 = 1 + 1 - 1 + 3 + 1 + 5 + 4= 14

S6 = 1 - 1 + 2 + 9 + 1 + 4 + 3 = 19

S7 = 1 - 1 + 3 + 3 + 4 - 1 - 1 = 8

S8 = 1 + 1 + 1 - 1 + 1 + 1 + 2 = 6

S9 = 1 + 1 - 1 + 3 + 1 + 3 + 1 = 9

S10 = 1 - 1 - 1 - 1 + 3 + 3 + 1 = 5

Les sessions seront proposées à l'utilisateur selon leurs scores dans l'ordre décroissant. Dans notre exemple, les sessions S3, S5 et S6 sont plus susceptibles de répondre au besoin de l'acteur-métier.

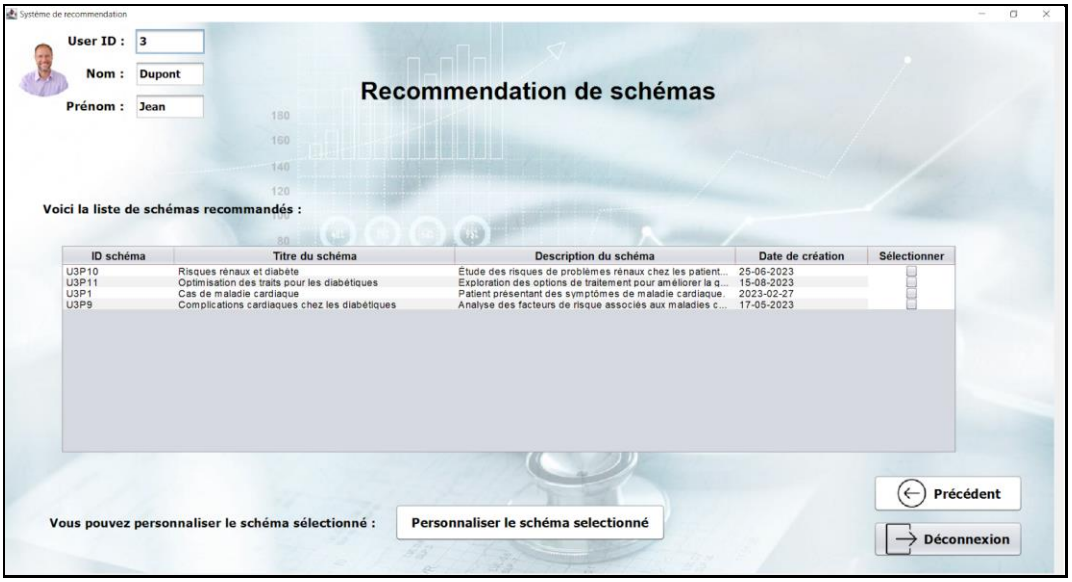


Figure 5.27 – Résultat de recommandation de schémas

Grâce à cette liste, l'utilisateur choisit le schéma qu'il considère le plus approprié. Il peut l'utiliser en l'état ou le personnaliser.

5.5. Validation

L'expérimentation du prototype DLtoDM que nous avons présentée dans les sections précédentes a porté sur une application médicale introduite dans le

chapitre 1 (cf. section 1.2). Elle a permis de montrer la faisabilité de nos propositions. L'objectif de nos travaux est bien d'assister les acteurs-métier dans la manipulation des données complexes contenues dans un LD. Ainsi, grâce au développement logiciel de notre solution, un acteur-métier se verra proposer soit une liste de schémas de magasin déjà élaborés soit la possibilité de créer un schéma. A partir de là, DLtoDM charge automatiquement les données dans le magasin. L'utilisateur est seul à être impliqué tout au long du processus d'élaboration. Cette autonomie permet à l'acteur-métier de construire son magasin rapidement (sans intermédiaire) au moment qu'il juge opportun.

A présent, il convient d'évaluer l'apport de notre solution dans un environnement professionnel.

Travaillant au sein de la société TRIMANE, nous avons pu obtenir la mise en œuvre de notre logiciel sur une application réelle (pour des raisons de confidentialité, certains noms des fichiers et des objets manipulés ont été changés). Nous avons mené une validation de nos propositions à travers l'application juridique "Météo du Droit". Cette application vise à mettre à disposition des juristes (les avocats et les notaires), la jurisprudence qui regroupe les décisions des tribunaux français et européens. La jurisprudence est composée d'importants volumes de données textuelles, comprenant notamment des arrêts du Conseil d'État, des décisions de tribunaux, des avis contentieux, des publications spécialisées, des revues d'analyse. Ces documents sont généralement stockés dans divers fonds documentaires numériques caractérisés par leur hétérogénéité et la complexité des croisements des données. Les données traitées, dont le volume atteint plusieurs téraoctets, sont stockées dans des BD relationnelles (MySQL et Microsoft Server SQL) et NoSQL (MongoDB). Des données sont également enregistrées dans des fichiers texte de type XML ; mais ceux-ci se situant hors de notre domaine d'étude, nous ne les considérons pas pour la validation. L'application a pour finalité de permettre aux utilisateurs d'interroger les connaissances juridiques stockées, en optimisant l'efficacité de ces opérations.

Afin d'évaluer la pertinence de l'approche, nous avons présenté le logiciel DLtoDM à des responsables et des acteurs de l'entreprise ayant un statut de décideur et n'ayant pas une expertise en informatique. Nous avons impliqué des participants ayant des profils variés en ce qui concerne leurs expériences. Sept participants ont réalisé les expérimentations qui leur ont été proposées. Nous leur avons demandé de tester DLtoDM pour élaborer des schémas décisionnels en vue d'accéder aux données d'un LD.

Nous avons utilisé le critère de temps de réponse du système pour comparer les résultats obtenus en mettant en œuvre successivement :

- le temps nécessaire pour interroger les BD du LD

- le temps mis par chaque décideur pour élaborer un schéma de magasin.

5.5.1. Le lac de données actuel

Dans l'application existante, il n'existe pas de LD explicite tel que nous l'avons défini. Cependant, les acteurs-métier sont conduits à poser des requêtes portant sur plusieurs ensembles de données ; nous verrons comment ils procèdent dans la section suivante. Dans un premier temps, nous allons décrire les données utilisées dans l'application juridique « Météo du Droit ».

Depuis plusieurs années, la France est engagée dans une dynamique d'ouverture des données détenues par les acteurs publics (Open data). L'accès en ligne aux décisions de justice est prévu par les textes depuis le décret du 7 août 2002 relatif au service public de la diffusion du droit par l'internet. C'est dans ce cadre que des données juridiques sont diffusées en libre accès.

Les praticiens du droit et les justiciables peuvent aujourd'hui consulter une sélection de décisions de tribunaux (jurisprudence) dans différents degrés de juridictions. La DILA (Direction pour l'Information Légale et Administrative) met à disposition différents jeux de données sur le site Légifrance :

- CAPP : Le jeu de données CAPP contient une sélection de décisions en matière civile et pénale des Cours d'appel et des juridictions de premier degré. Anonymisées, ces décisions sont composées du texte intégral des arrêts complétées, pour 17 % d'entre elles, par des titrages et des sommaires rédigés par les Cours d'appel en liaison avec le service de documentation et d'études de la Cour de cassation.
- CASS : Le jeu de données CASS contient les arrêts de la Cour de cassation qui sont publiés aux :
    - Bulletin des chambres civiles depuis 1960
    - Bulletin de la chambre criminelle depuis 1963.

Hebdomadairement, les décisions sont mises à jour par la Cour de cassation.

- INCA : Le jeu de données INCA contient les arrêts inédits (non publiés au Bulletin) diffusés par le fond de concours de la Cour de cassation depuis 1989. Anonymisées, ces décisions sont composées du texte intégral des arrêts.
- CONSTIT : Le jeu de données CONSTIT contient toutes les décisions rendues par le Conseil constitutionnel depuis sa création par la Constitution du 4 octobre 1958 (contrôle de constitutionnalité,

contentieux électoral, nominations et autres décisions). Ces décisions sont composées du texte intégral. Pour celles qui relèvent du contrôle de constitutionnalité des lois, elles sont composées de lettres de saisine et d'observations du gouvernement. Elles sont fournies en fonction de l'activité juridictionnelle du Conseil constitutionnel.

- JADE : Le jeu de données JADE contient les décisions du Conseil d'Etat, des cours administratives d'appel et du tribunal. Anonymisées, ces décisions sont composées du texte intégral des décisions. Les décisions publiées au « Recueil Lebon » sont complétées par des résumés rédigés par le centre de documentation du Conseil d'Etat.

La DILA met à disposition d'autres jeux de données en open data sur Légifrance mais « La Météo du Droit » s'appuie pour l'instant uniquement sur un LD dont dispose TRIMANE. Ce LD constitué de l'ensemble des jeux de données juridiques évoqués précédemment (CAPP, CASS, INCA, CONSTIT et JADE). Au total nous avons recensé 1 006 045 décisions de justice. Nous présentons ci-dessous des extraits de trois BD « CasesDB », « TrialsDB » et « JudgmentsDB » du LD « Météo du Droit ». Ces BD ont été élaborées par TRIMANE pour tester l'application « Météo du Droit » et décrivent respectivement des informations sur des affaires judiciaires, des données liées aux procès judiciaires et des données dédiées aux décisions judiciaires (cf. figures 5.28, 5.29 et 5.30).

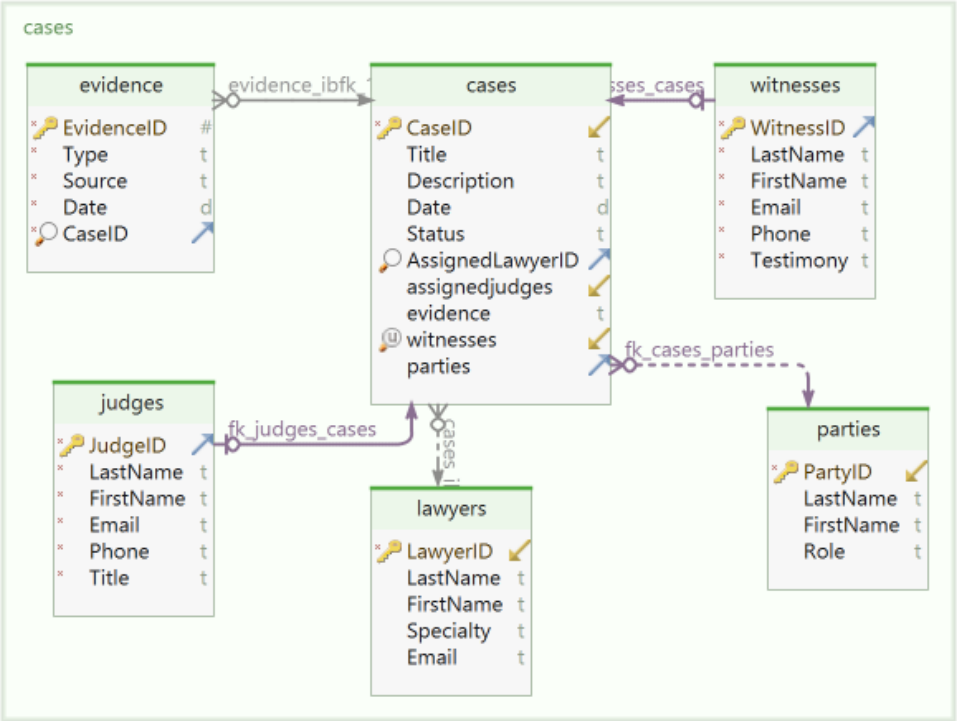


Figure 5.28 – Extrait du schéma de la BD relationnelle « CasesDB »

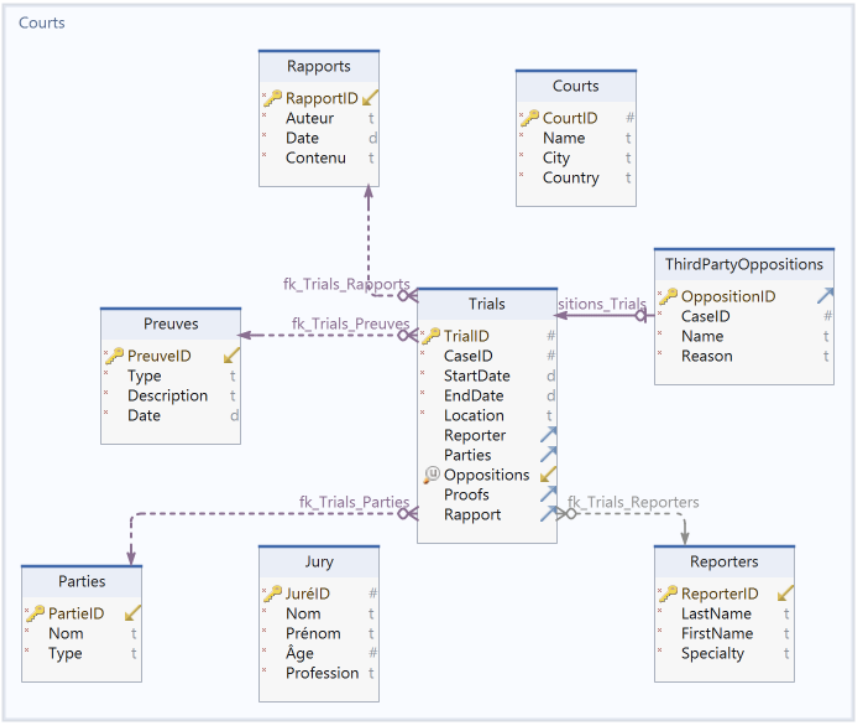


Figure 5.29 – Extrait du schéma de la BD relationnelle « TrialsDB »

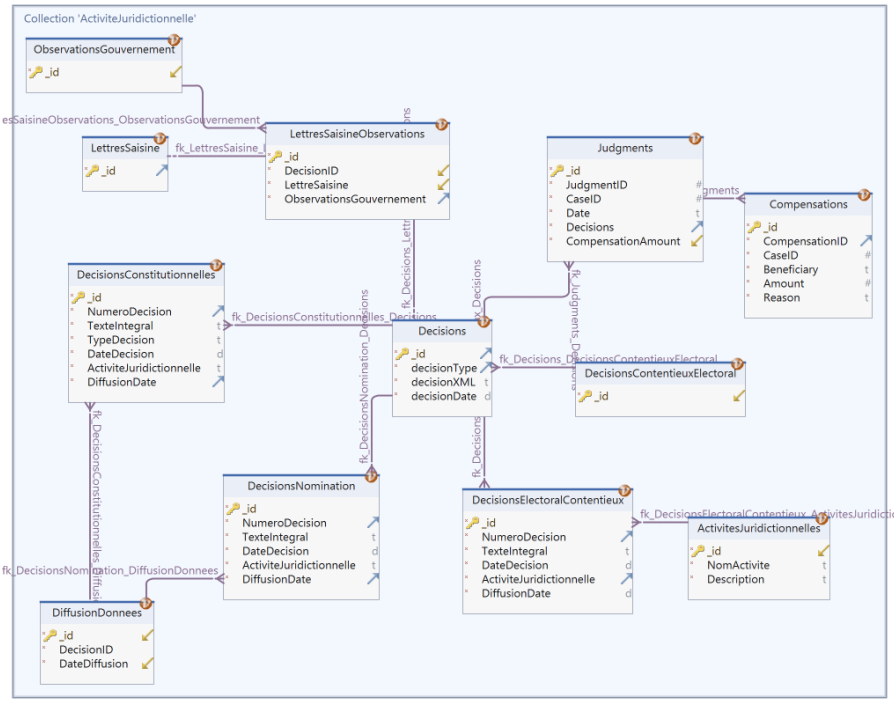


Figure 5.30 – Extrait de la BD MongoDB « JudgmentsDB »

5.5.2. Déroulement actuel des traitements

Dans notre contexte, les acteurs-métier doivent exprimer des requêtes sur des ensembles de données distincts et hétérogènes, pouvant être plus ou moins corrélés. Ces données étant distribuées sur plusieurs supports, il serait alors envisageable de recourir à des techniques développées dans le domaine des BD réparties. Mais le volume et la variété de ces ensembles de données sont tels que les techniques de BD réparties (BDR unifiées ou multibases) sont difficilement applicables. Face à ces contraintes du Big data, il est préalablement nécessaire d'extraire les données utiles aux acteurs-métier puis de les centraliser et de les homogénéiser avant de les traiter.

Nous avons demandé à sept acteurs-métier de réaliser dix requêtes sur les trois BD tests. Ces requêtes sont données dans le tableau 5.5 suivant et classées par ordre croissant de difficulté :

Requêtes des acteurs-métier
Requête 1 : "Obtenir les décisions de justice récentes rendues au cours des 2 dernières années relatives aux litiges de droits d'auteur dans l'industrie du divertissement notamment les secteurs de la musique, du cinéma et de la télévision. Fournir les détails des cas, les parties impliquées, les décisions prises par les tribunaux et les montants de compensation éventuellement accordés."
Requête 2 : "Trouver des cas judiciaires liés à la discrimination fondée sur le genre au cours des 5 dernières années. Inclure les décisions de justice qui ont établi des précédents importants en matière de lutte contre la discrimination de genre. Présenter les détails des cas, les parties concernées, les arguments avancés, les jugements rendus et les mesures correctives appliquées, le cas échéant."
Requête 3 : "Fournir des détails sur les accidents de la route survenus au cours des 2 dernières années. Inclure les informations sur les parties impliquées, les circonstances de l'accident, les dommages matériels et corporels, ainsi que les jugements rendus par les tribunaux."
Requête 4 : "Rechercher des cas de harcèlement au travail rapportés au cours des 3 dernières années. Présenter les détails des cas, y compris les témoignages, les preuves collectées, les mesures disciplinaires prises et les verdicts rendus par les tribunaux."
Requête 5 : "Trouver des décisions de justice clés concernant les limites de la liberté d'expression dans le contexte numérique. Présenter les cas emblématiques, les arguments des parties, les critères juridiques appliqués et les jugements rendus par les tribunaux."

Requête 6 : "Identifier les litiges contractuels impliquant des entreprises, survenus au cours des 2 dernières années. Présenter les entreprises concernées, les clauses contractuelles litigieuses, les négociations, les solutions juridiques trouvées et les décisions rendues par les tribunaux."
Requête 7 : "Rechercher des affaires médicales où la responsabilité professionnelle des praticiens a été mise en cause. Inclure les détails des cas, les allégations de négligence, les expertises médicales, les témoignages d'experts et les verdicts rendus par les tribunaux."
Requête 8 : "Trouver des cas de litiges environnementaux récents impliquant des questions de pollution ou de préservation des ressources naturelles. Présenter les parties en conflit, les préoccupations environnementales, les preuves présentées, les mesures compensatoires et les verdicts rendus par les tribunaux."
Requête 9 : "Identifier les affaires judiciaires impliquant des litiges entre États au cours des 5 dernières années. Présenter les pays concernés, les enjeux diplomatiques, les arguments juridiques avancés, les décisions de la Cour internationale de Justice et les éventuelles résolutions adoptées."
Requête 10 : "Trouver les détails des cas de litiges en droit de la propriété intellectuelle incluant les décisions prises par les tribunaux et les montants de compensation accordés. Afficher également les noms des avocats assignés à ces cas ainsi que les rapporteurs impliqués dans les procès."

Tableau 5.5 – Requêtes tests

Actuellement, pour exprimer une requête sur les BD existantes, un décideur la décompose manuellement en sous-requêtes, chacune d'elles portant sur une BD unique. Ensuite, des opérations ensemblistes sont appliquées sur les résultats (généralement l'union) pour obtenir le résultat de la requête initiale.

Prenons l'exemple de la requête n°10 : "Trouver les détails des cas de litiges en droit de la propriété intellectuelle …". Sa décomposition en sous-requêtes est donnée dans le tableau 5.6.

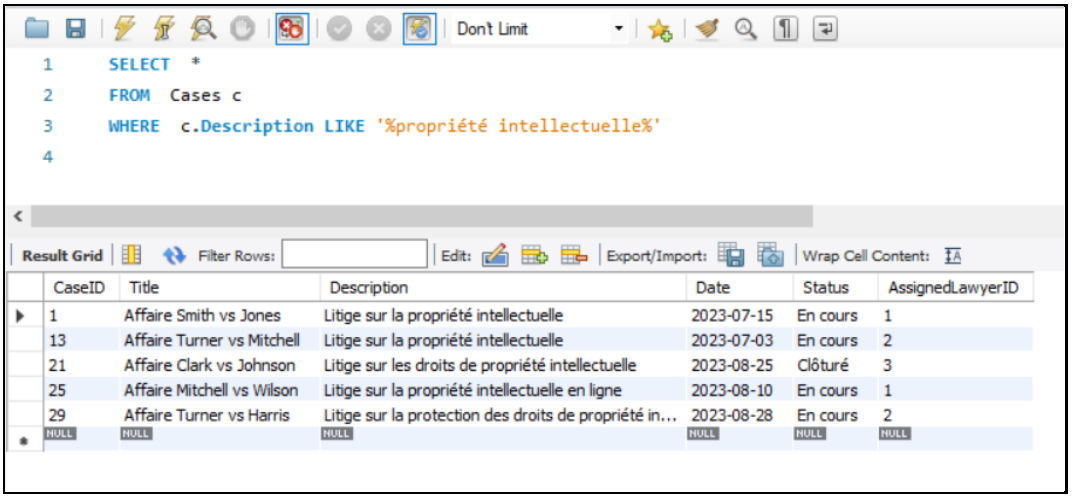
Num	Sous-requête	BD concernée	Traduction de la requête
1	Trouver les cas de litiges en droit de la propriété	CasesDB	SELECT * FROM Cases c WHERE c.Description LIKE '%propriété intellectuelle%'

	intellectuelle avec des jugements		
2	Récupérer les détails des jugements et compensations associées	JudgmentsDB	db.getCollection("Judgments").find({ CaseID: { $in: [1, 13, 21, 25, 29] } }, { JudgmentID: 1, Date: 1, Decisions: 1, CompensationAmount: 1, CaseID: 1, _id: 0 });
3	Obtenir les noms des avocats assignés aux cas	CasesDB	SELECT * FROM Lawyers WHERE LawyerID IN (SELECT AssignedLawyerID FROM Cases)
4	Retrouver les rapporteurs impliqués dans les procès	TrialsDB	SELECT [ReporterID], [LastName], [FirstName] FROM Reporters INNER JOIN Trials ON Reporters.ReporterID = Trials.Reporter WHERE Trials.CaseID in (1, 13, 21, 25, 29) ;

Tableau 5.6 – Décomposition de la requête n°10

Dans ce qui suit, nous montrons les résultats de chacune de ces sous-requêtes.

Pour les trois sous-requêtes relationnelles, les résultats de leurs exécutions sous MySQL et MSSQL sont respectivement illustrés dans les figures 5.31, 5.32 et 5.33 suivantes.



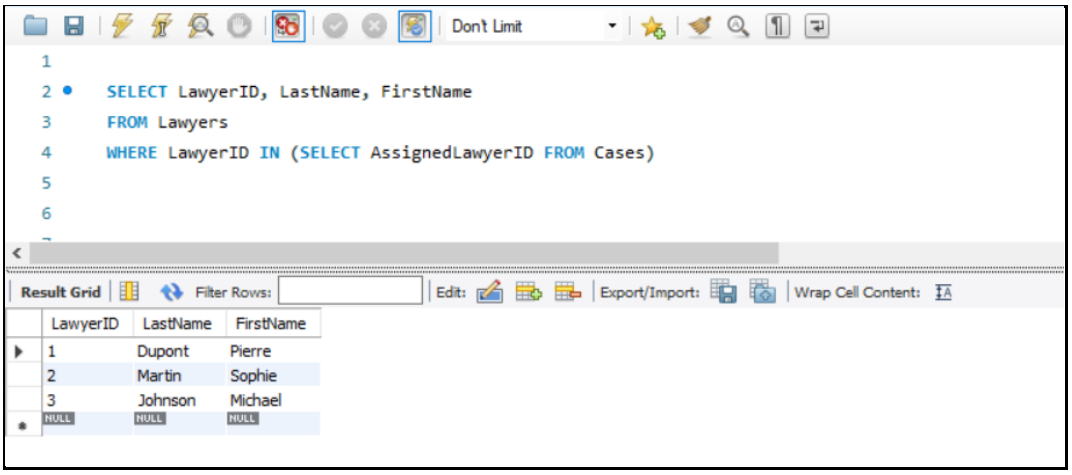
```
SELECT *
FROM Cases c
WHERE c.Description LIKE '%propriété intellectuelle%'
```

Result Grid

CaseID	Title	Description	Date	Status	AssignedLawyerID
1	Affaire Smith vs Jones	Litige sur la propriété intellectuelle	2023-07-15	En cours	1
13	Affaire Turner vs Mitchell	Litige sur la propriété intellectuelle	2023-07-03	En cours	2
21	Affaire Clark vs Johnson	Litige sur les droits de propriété intellectuelle	2023-08-25	Clôturé	3
25	Affaire Mitchell vs Wilson	Litige sur la propriété intellectuelle en ligne	2023-08-10	En cours	1
29	Affaire Turner vs Harris	Litige sur la protection des droits de propriété in...	2023-08-28	En cours	2

Figure 5.31 – Résultat de la sous-requête n°1

Ici, les numéros des cas de litiges « CaseID » vont être utilisés dans les sous-requêtes suivantes.



```
SELECT LawyerID, LastName, FirstName
FROM Lawyers
WHERE LawyerID IN (SELECT AssignedLawyerID FROM Cases)
```

Result Grid

LawyerID	LastName	FirstName
1	Dupont	Pierre
2	Martin	Sophie
3	Johnson	Michael

Figure 5.32 – Résultat de la sous-requête n°3

```
SELECT distinct [CaseID], [ReporterID] ,[LastName] ,[FirstName]
  FROM Reporters
  INNER JOIN Trials
  ON Reporters.ReporterID = Trials.Reporter
  WHERE Trials.CaseID in (1, 13, 21, 25, 29)
```

100 %

Résultats | Messages

	CaseID	ReporterID	LastName	FirstName
1	1	1	Dupond	Jacques
2	13	10	Martin	Catherine
3	21	11	Dupont	Michel
4	25	15	Dupont	François
5	29	16	Martin	Isabelle

Figure 5.33 – Résultat de la sous-requête n°4

De la même façon, la sous-requête n°2 sur MongoDB utilise les numéros de cas de litiges résultant de la sous-requête n°1. Le résultat est illustré à la figure 5.34.


```
> db.getCollection("Judgments").find(
...     { CaseID: { $in: [1, 13, 21, 25, 29] } },
...     { JudgmentID: 1, Date: 1, Decisions: 1, CompensationAmount: 1, CaseID: 1, _id: 0 }
... );
{ "JudgmentID" : 1, "CaseID" : 1, "Date" : "2023-09-15", "Decisions" : "La propriété intellectuelle est accordée au plaignant", "CompensationAmount" : 50000 }
{ "JudgmentID" : 2, "CaseID" : 13, "Date" : "2023-09-20", "Decisions" : "Violation de brevet établie.", "CompensationAmount" : 25000 }
{ "JudgmentID" : 3, "CaseID" : 21, "Date" : "2023-07-05", "Decisions" : "Litige résolu en faveur du plaignant.", "CompensationAmount" : 15000 }
{ "JudgmentID" : 4, "CaseID" : 25, "Date" : "2023-08-25", "Decisions" : "Dommages et intérêts accordés en cas de violation.", "CompensationAmount" : 18000 }
{ "JudgmentID" : 5, "CaseID" : 29, "Date" : "2023-09-05", "Decisions" : "Violation de droits d'auteur confirmée.", "CompensationAmount" : 12000 }
>
```


Figure 5.34 – Résultat de la sous-requête n°2

Le décideur collecte les résultats des sous-requêtes dans une structure de données comme une feuille de calcul Excel (cf. figure 5.35).

	A	B	C	D	E	F	G	H	I	J
1	CaseID	Titre	Description	DateLitige	Status	Decision	DateDecision	Montant	ReporetrID	LawyerID
2	1	Affaire Smith vs Jones	Litige sur la propriété intellectuelle	15/07/2023	En cours	La propriété intellectuelle est accordée au plaignant	15/09/2023	50000	1	1
3	13	Affaire Turner vs Mitchell	Litige sur la propriété intellectuelle	03/07/2023	En cours	Violation de brevet établie.	20/09/2023	25000	10	2
4	21	Affaire Clark vs Johnson	Litige sur les droits de propriété intellectuelle	25/08/2023	Clôturé	Litige résolu en faveur du plaignant.	05/07/2023	15800	11	3
5	25	Affaire Mitchell vs Wilson	Litige sur la propriété intellectuelle en ligne	10/08/2023	En cours	Dommages et intérêts accordés en cas de violation.	25/08/2023	10000	15	1
6	29	Affaire Turner vs Harris	Litige sur la protection des droits de propriété intellectuelle	28/08/2023	En cours	"Violation de droits d'auteur confirmée	05/09/2023	12000	16	2

Figure 5.35 – Union des résultats de sous-requêtes dans un fichier

L'union des résultats de ces sous-requêtes répond à la requête initialement posée par le décideur et donne des informations sur les détails des cas de litiges en droit de la propriété intellectuelle en interrogeant les trois BD. Ce traitement manuel est lent et fastidieux.

5.5.3. Traitements avec utilisation de DLtoDM

Dans cette section, nous présentons successivement la mise en œuvre des deux processus constituant notre système.

Le processus DLtoDW permet de centraliser les données contenues dans plusieurs BD en les stockant dans un entrepôt. Une description de ce mécanisme est fournie dans le chapitre 3. Notre processus est automatisé et basé sur MDA (cf. section 5.3.3). Nous avons utilisé le SGBD OrientDB pour implanter l'entrepôt.

Le temps mis pour élaborer le même entrepôt à partir de BD choisies par l'utilisateur a pris environ 11 minutes. Autrement dit, la centralisation des données des BD réalisées avec notre processus permet un gain de temps significatif par rapport au processus manuel ; ceci est montré dans le tableau 5.7 ci-dessous.

Les acteurs-métier choisis n'avaient pas, jusqu'alors, travaillé sur l'application juridique ; ils ne connaissaient pas les schémas des BD stockées dans le LD. Nous avons donc demandé à un second décideur d'exécuter la même requête n°10 en utilisant l'entrepôt résultant du processus DLtoDW. Il a utilisé dans un premier temps, le mécanisme de personnalisation pour élaborer son schéma de magasin et répondre à la requête).

Nous avons comparé le temps nécessaire à notre processus automatisé avec le temps mis par le décideur pour poser une requête en utilisant la méthode manuelle actuellement utilisée (détaillée dans la section précédente). Pour la requête n°10, le décideur met environ 25 minutes pour poser et exécuter la requête manuellement. Nous avons comparé cette durée à la durée mise par le mécanisme de personnalisation pour obtenir les mêmes résultats (cf. tableau 5.7).

	Temps mis pour l'interrogation des BD
Méthode manuelle	25 minutes
Méthode automatique (DWtoDM)	11 minutes

Tableau 5.7 – Temps de réalisation des processus manuel et automatisé

Dans un second temps, la même requête posée est reprise par un 3ème décideur pour tester le mécanisme de recommandation. Ce dernier lui a

proposé des schémas de magasin. Le schéma proposé est comparé au schéma élaboré par le décideur via le mécanisme de personnalisation.

Requête du décideur	Schéma proposé	Validation du décideur
Requête 10	- Classe "Case": CaseID, Title, Description, Date, Status - Classe "Lawyer": LawyerID, LastName, FirstName, Specialty, Email - Classe "Judgment": JudgmentID, Date, Decisions, CompensationAmount - Classe "Reporter":ReporterID, LastName, FirstName, Specialty	Oui

Tableau 5.8 – Schéma proposé par le système

Concernant le mécanisme de recommandation, d'après le tableau 5.8, le schéma proposé répond bien aux besoins du décideur. Ceci devrait contribuer à améliorer l'efficacité de la prise de décision dans les entreprises.

5.6. Conclusion du chapitre

Dans ce chapitre, nous avons décrit l'implantation des deux processus de notre système : l'ingestion des données du LD (DLtoDW) et l'accès aux données via la personnalisation ou la recommandation de schémas de magasin (DWtoDM).

Le processus d'ingestion des données DLtoDW proposé comprend trois modules qui sont mis en œuvre successivement. Tout d'abord, le module *Transfer* qui extrait les données sélectionnées à partir des BD du LD et les charge dans un entrepôt. Ce module repose sur l'architecture MDA. Le module *Merging* fusionne les classes considérées sémantiquement similaires. Cette fusion repose sur l'ontologie « Onto » alimentée par des experts-métiers. Enfin, le module *Convert* traduit les liens relationnels et NoSQL sous la forme de références conformément aux principes des BD objet.

Le processus DWtoDM a pour objectif d'assister les acteurs-métier pour accéder aux données complexes de l'entrepôt. Il permet la création de schémas de magasin à partir de requêtes en langage naturel.

Pour tester le prototype, nous avons mené des expérimentations qui portent sur une application médicale (présentée au chapitre 1). Pour ceci, nous avons instancié les métamodèles des BD proposés dans le chapitre 3. Une suite de règles de transformations appliquées à ces méta modèles a permis de générer l'entrepôt. Ensuite, notre système d'assistance détaillé au niveau du chapitre 4 a permis l'accès aux données complexes de l'entrepôt.

Ce chapitre montre également la pertinence de nos propositions à travers une validation. Celle-ci a été effectuée dans le cadre d'un projet réalisé par la société TRIMANE. L'objectif est de comparer le temps de réalisation de notre processus automatisé avec le temps mis par un utilisateur pour obtenir, sans l'aide de notre système, un résultat équivalent (entrepôt NoSQL et des schémas de magasins).

CONCLUSION ET PERSPECTIVES

Rappel des verrous et bilan des contributions

Les travaux présentés dans ce mémoire de thèse se situent dans le contexte des LD et portent plus précisément sur l'accès à des mégadonnées stockées sur un SGBD NoSQL.

Nos travaux ont été motivés par une problématique métier de la société TRIMANE et concerne la gestion de données complexes pour les mutuelles santé et les centres médicaux. Cette problématique repose sur le besoin d'exploiter des mégadonnées stockées dans un LD médical. Parmi l'ensemble des problèmes rencontrés, nous avons focalisé nos travaux sur deux verrous. Le premier consiste à réorganiser les données complexes contenues dans un LD pour les stocker dans une BD NoSQL cible. Le second concerne l'accès à ces données par des acteurs-métier sans connaissances en informatique préalables.

En réponse à cette problématique, nous avons proposé une solution permettant à des acteurs-métier de concevoir et mettre en œuvre, de manière autonome, des magasin élaborés à partir d'un LD. Nos contributions s'articulent selon deux parties : celles sur la conception, la modélisation et l'implantation d'un entrepôt NoSQL, dans le but de stocker de manière centralisée les données hétérogènes extraites du LD, et celles sur la définition d'un mécanisme de personnalisation et de recommandation pour la création de schémas de magasin à partir de l'entrepôt. Ces deux parties sont détaillées ci-dessous.

Les travaux de cette thèse ont été présentés dans les publications suivantes : [9], [10], [11], [12] et [13].

Contributions à la création d'un entrepôt à partir d'un LD

Face à l'hétérogénéité des BD contenues dans le LD, il convient de réorganiser ces données sous une forme homogène, en prenant en charge la fusion des données "similaires" ainsi que la conversion des liens sous la forme de références entre objets.

Nous avons donc proposé le processus DLtoDW composé de trois modules qui assurent successivement le transfert des données sources du LD vers l'entrepôt (module *Transfer*), la fusion des objets dits "similaires" (module *Merging*) et la conversion des liens sémantiques dans la cible (module *Convert*). Pour ce faire, nous avons proposé une solution basée sur l'architecture MDA en formalisant notre processus et en le généralisant à

plusieurs sources de données décrites ; ce qui assure une certaine généricité de notre processus.

Contributions à la création de schémas de magasin

Notre seconde contribution concerne l'accès aux données complexes de l'entrepôt. Nous avons proposé, aux acteurs-métier, des mécanismes simples permettant de transformer leurs besoins de données en un schéma de magasins. Deux mécanismes sont mis en œuvre pour assurer la sélection des données et par la suite la création de schémas de magasins. L'acteur-métier a le choix :

- soit de modifier un schéma qu'il a précédemment créé (mécanisme de personnalisation).
- soit de demander des recommandations de schémas élaborés par d'autres utilisateurs ayant des comportements similaires. Il exprime ainsi son besoin sous la forme d'une phrase en langage naturel et le système lui retourne une liste de schémas susceptibles de répondre à son besoin (mécanisme de recommandation).

Ensuite, durant le processus de requêtage du magasin (hors champ d'étude de cette thèse), des métadonnées sont stockées pour assurer et contribuer au processus de recommandation.

Expérimentation et validation

Afin de valider nos propositions, nous avons développé un logiciel dénommé DLtoDM qui est composé de deux processus DLtoDW et DWtoDM exécutés successivement. Le module *Transfer* de DLtoDW est développé en utilisant la plateforme EMF (Eclipse Modeling Framework) [90]. Cette plateforme d'implantation présente un environnement complet et adapté à la métamodélisation, la modélisation et la transformation des modèles. Elle dispose de l'ensemble d'outils nécessaires à la mise en œuvre de notre solution MDA. Les modules *Merging* et *Convert* reposent sur des algorithmes mis en œuvre en JAVA.

La second processus DWtoDM facilite la création de magasin à partir d'un entrepôt par un mécanisme de personnalisation ou de de recommandation en interagissant avec l'acteur-métier. Ce dernier a la possibilité de faire appel à un tel mécanisme de pour obtenir des schémas de magasin élaborés précédemment par des acteurs-métier ayant des profils similaires. Le module de recommandation comporte plusieurs étapes. Tout d'abord, les besoins exprimés par l'utilisateur sont analysés et convertis en attributs présents

dans l'entrepôt. Ensuite, une pondération des attributs est effectuée selon leur présence dans le schéma. Enfin, le calcul du score final permet de classer les schémas de magasin recommandés en fonction de leur pertinence par rapport aux besoins. Le module de personnalisation permet à l'utilisateur de créer son propre magasin en se basant sur le schéma de l'entrepôt global.

Pour conclure, les travaux menés au cours de cette thèse apportent des contributions à la thématique de l'exploitation des LD dans un cadre décisionnel et répondent à un besoin métier. Mais ces contributions s'accompagnent aussi de nouvelles perspectives.

Perspectives de recherche

Nous envisageons de poursuivre nos travaux. Jusqu'à présent, nous avons apporté une contribution à ce projet à deux niveaux :

La définition d'une démarche et la réalisation d'un prototype permettant l'ingestion de données d'un LD contenant des sources hétérogènes (relationnelles et NoSQL) dans un entrepôt.

La définition d'une démarche et la réalisation d'un prototype permettant aux acteurs-métier de créer des schémas de magasin issus de l'entrepôt. Ceci offre aux acteurs-métier la possibilité de construire les schémas de magasin en fonction de leurs besoins spécifiques, tout en bénéficiant d'un processus simplifié et guidé.

Dans le prolongement de ces travaux, nous envisageons d'intégrer d'autres types de sources de données dans le système DLtoDM. Actuellement, notre système intègre des sources relationnelles et NoSQL orientées-documents. Nous souhaitons permettre aux acteurs-métier d'intégrer, au sein du même entrepôt, des fichiers JSON et XML, des fichiers plats ou des données provenant de BD NoSQL de types autres que ceux déjà étudiés (orientées-colonnes, orientées-graphes). Ceci contribuera à une plus grande flexibilité et à une gestion plus complète des données dans le système DLtoDM. Ceci nécessite typiquement (1) la modélisation de ces types de sources de données, puis (2) la mise à jour de l'ensemble des métamodèles et des règles de transformation utilisées.

Le deuxième point concerne l'industrialisation de DLtoDM implique également la mise en place de processus standardisés et reproductibles pour l'ingestion d'un LD dans un entrepôt. Cela permettra d'automatiser les flux de travail et de réduire les efforts manuels, tout en assurant la cohérence et la qualité des données. En outre, la modularisation de DLtoDM est envisageable et permettra de concevoir le système sous forme de modules distincts, ce qui facilitera la gestion, la maintenance et l'évolutivité de l'architecture. Chaque module pourra être développé, testé et déployé indépendamment, ce qui

offrira une plus grande agilité dans l'ajout de nouvelles sources de données et dans les évolutions du système.

Le troisième point est lié à l'amélioration de l'expérience utilisateur. Ainsi, nous proposons d'intégrer un agent conversationnel basé sur l'Intelligence Artificielle (IA) générative dans le système DLtoDM. Cet agent conversationnel agit comme une interface intuitive permettant d'accéder aux données du LD. Les utilisateurs pourront interagir avec l'agent conversationnel en posant des questions et en demandant des recommandations sur les BD à extraire du LD puis des recommandations sur les données de l'entrepôt NoSQL créé. L'agent conversationnel utilisera des techniques d'IA générative pour comprendre les requêtes des utilisateurs, générer des réponses appropriées et fournir une expérience utilisateur améliorée et plus ciblée. Ceci facilitera l'exploitation des données du LD, en rendant l'interaction plus conviviale pour les utilisateurs.

Bibliographie

[1] D. R.-J. G.-J. Rydning, J. Reinsel, et J. Gantz, « The digitization of the world from edge to core », *Framingham: International Data Corporation*, vol. 16, p. 1-28, 2018.

[2] W. H. Inmon, D. Strauss, et G. Neushloss, *DW 2.0: The Architecture for the Next Generation of Data Warehousing*. Elsevier, 2010.

[3] R. Kimball et M. Ross, *The data warehouse toolkit: the definitive guide to dimensional modeling*. John Wiley & Sons, 2013.

[4] P. P. Khine et Z. S. Wang, « Data lake: a new ideology in big data era », in *ITM web of conferences*, EDP Sciences, 2018, p. 03025.

[5] A. LaPlante, *Architecting data lakes: data management architectures for advanced business use cases*, First edition. Sebastopol, CA: O'Reilly Media, 2016.

[6] H. Chen, R. H. Chiang, et V. C. Storey, « Business intelligence and analytics: From big data to big impact », *MIS quarterly*, p. 1165-1188, 2012.

[7] J. Dixon, « Pentaho, hadoop, and data lakes ». 2010.

[8] P. N. Sawadogo, E. Scholly, C. Favre, E. Ferey, S. Loudcher, et J. Darmont, « Metadata systems for data lakes: models and features », in *New Trends in Databases and Information Systems: ADBIS 2019 Short Papers, Workshops BBIGAP, QAUCA, SemBDM, SIMPDA, M2P, MADEISD, and Doctoral Consortium, Bled, Slovenia, September 8–11, 2019, Proceedings 23*, Springer, 2019, p. 440-451.

[9] F. Abdelhedi, R. Jemmali, et G. Zurfluh, « Ingestion of a Data Lake into a NoSQL Data Warehouse: The Case of Relational Databases. », in *13th International Conference on Knowledge Management and Information Systems (KMIS 2021)*, 2021, p. 64-72.

[10] F. Abdelhedi, R. Jemmali, et G. Zurfluh, « Relational Databases Ingestion into a NoSQL Data Warehouse », présenté à ICSEA 2021, The Sixteenth International Conference on Software Engineering Advances, oct. 2021, p. 128-133.

[11] F. Abdelhedi, R. Jemmali, et G. Zurfluh, « Data Ingestion from a Data Lake: The Case of Document-oriented NoSQL Databases », in *24th International Conference on Enterprise Information Systems (ICEIS 2022)*, SCITEPRESS: Science and Technology Publications, 2022, p. 226-233.

[12] F. Abdelhedi, R. Jemmali, et G. Zurfluh, « DLtoDW: Transferring relational and NoSQL databases from a data lake », *Springer Nature Computer Science*, vol. 3, n° 5, p. 381, 2022.

[13] F. Abdelhedi, R. Jemmali, et G. Zurfluh, « Medical data lake query assistance », in *20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2023)*, Caire, Egypt: IEEE, déc. 2023, p. à paraître.

[14] C. Arthur, « Tech giants may be huge, but nothing matches big data », *The Guardian*, vol. 23, 2013.

[15] E. F. Codd, « Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate », *http://www. arborsoft. com/papers/coddTOC. html*, 1993.
[16] O. Teste, « Modélisation et manipulation d'entrepôts de données complexes et historisées », PhD Thesis, Université Paul Sabatier-Toulouse III, 2000.
[17] C. Madera, « L'évolution des systèmes et architectures d'information sous l'influence des données massives: les lacs de données », PhD Thesis, Université Montpellier, 2018.
[18] R. Kimball et M. Ross, *The data warehouse toolkit: the definitive guide to dimensional modeling*. John Wiley & Sons, 2013.
[19] W. H. Inmon, D. Strauss, et G. Neushloss, *DW 2.0: The architecture for the next generation of data warehousing*. Elsevier, 2010.
[20] I. A. Ajah et H. F. Nweke, « Big data and business analytics: Trends, platforms, success factors and applications », *Big Data and Cognitive Computing*, vol. 3, nº 2, p. 32, 2019.
[21] K. Krishnan, *Data Warehousing in the Age of Big Data*. Newnes, 2013.
[22] N. Miloslavskaya et A. Tolstoy, « Big data, fast data and data lake concepts », *Procedia Computer Science*, vol. 88, p. 300-305, 2016.
[23] Y. Riahi et S. Riahi, « Big data and big data analytics: Concepts, types and technologies », *International Journal of Research and Engineering*, vol. 5, nº 9, p. 524-528, 2018.
[24] A. Dhaouadi, K. Bousselmi, S. Monnet, M. Gammoudi, et S. Hammoudi, « A Multi-layer Modeling for the Generation of New Architectures for Big Data Warehousing », 2022, p. 204-218.
[25] K. Dehdouh, O. Boussaid, et F. Bentayeb, « Big Data Warehouse: Building Columnar NoSQL OLAP Cubes », *IJDSST*, vol. 12, nº 1, p. 1-24, janv. 2020.
[26] A. Fekete, « CAP Theorem », in *Encyclopedia of Database Systems*, L. Liu et M. T. Özsu, Éd., New York, NY: Springer, 2018, p. 395-396.
[27] E. Scholly, « De la modélisation des métadonnées à la conception d'un lac de données: Application à l'habitat social », PhD Thesis, Université de Lyon, 2022.
[28] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, et P. C. Arocena, « Data lake management: challenges and opportunities », *Proceedings of the VLDB Endowment*, vol. 12, nº 12, p. 1986-1989, 2019.
[29] O. Fennelly *et al.*, « Successfully implementing a national electronic health record: a rapid umbrella review », *International Journal of Medical Informatics*, vol. 144, p. 104281, 2020.
[30] S. Imran, T. Mahmood, A. Morshed, et T. Sellis, « Big data analytics in healthcare- A systematic literature review and roadmap for practical implementation », *IEEE/CAA Journal of Automatica Sinica*, vol. 8, nº 1, p. 1-22, 2020.
[31] M. Souibgui, F. Atigui, S. B. Yahia, et S. S.-S. Cherfi, « IRIS-DS: A new approach for identifiers and references discovery in document stores », in *54th Hawaii International Conference on System Sciences (HICSS 2021)*, HICSS, 2021, p. 970-979.

[32] F. Abdelhedi, H. Rajhi, et G. Zurfluh, « Extraction of Semantic Links from a Document-Oriented NoSQL Database », *SN COMPUT. SCI.*, vol. 4, n° 2, p. 148, janv. 2023.
[33] « DB-Engines Ranking », DB-Engines. [En ligne]. Disponible sur: https://db-engines.com/en/ranking/document+store
[34] « pivotal-business-data-lake-technical_brochure_web.pdf ». Consulté le: 10 septembre 2023. [En ligne]. Disponible sur: https://www.capgemini.com/wp-content/uploads/2017/07/pivotal-business-data-lake-technical_brochure_web.pdf
[35] B. Inmon, *Data Lake Architecture: Designing the Data Lake and avoiding the garbage dump*. Technics publications, 2016.
[36] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, et B. Mitschang, « Leveraging the data lake: Current state and challenges », in *Big Data Analytics and Knowledge Discovery: 21st International Conference, DaWaK 2019, Linz, Austria, August 26–29, 2019, Proceedings 21*, Springer, 2019, p. 179-188.
[37] R. Nadipalli, *Effective business intelligence with QuickSight*. Packt Publishing Ltd, 2017.
[38] P. Subramaniam, Y. Ma, C. Li, I. Mohanty, et R. C. Fernandez, « Comprehensive and comprehensible data catalogs: the what, who, where, when, why, and how of metadata management », *arXiv preprint arXiv:2103.07532*, 2021.
[39] P. Jovanovic, S. Nadal, O. Romero, A. Abelló, et B. Bilalli, « Quarry: a user-centered big data integration platform », *Information Systems Frontiers*, vol. 23, p. 9-33, 2021.
[40] J. C. Couto et D. D. Ruiz, « An overview about data integration in data lakes », in *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, IEEE, 2022, p. 1-7.
[41] A. Farrugia, R. Claxton, et S. Thompson, « Towards social network analytics for understanding and managing enterprise data lakes », in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, IEEE, 2016, p. 1213-1220.
[42] P. L. Giudice, L. Musarella, G. Sofo, et D. Ursino, « An approach to extracting complex knowledge patterns among concepts belonging to structured, semi-structured and unstructured sources in a data lake », *Information Sciences*, vol. 478, p. 606-626, 2019.
[43] C. Diamantini, P. L. Giudice, L. Musarella, D. Potena, E. Storti, et D. Ursino, « A new metadata model to uniformly handle heterogeneous data lake sources », in *New Trends in Databases and Information Systems: ADBIS 2018 Short Papers and Workshops, AI* QA, BIGPMED, CSACDB, M2U, BigDataMAPS, ISTREND, DC, Budapest, Hungary, September, 2-5, 2018, Proceedings 22*, Springer, 2018, p. 165-177.
[44] I. Lopez-Gazpio, M. Maritxalar, M. Lapata, et E. Agirre, « Word n-gram attention models for sentence similarity and inference », *Expert Systems with Applications*, vol. 132, p. 1-11, 2019.
[45] C. Quix, R. Hai, et I. Vatov, « Metadata extraction and management in data lakes with GEMMS », *Complex Systems Informatics and Modeling Quarterly*, n° 9, p. 67-83, 2016.

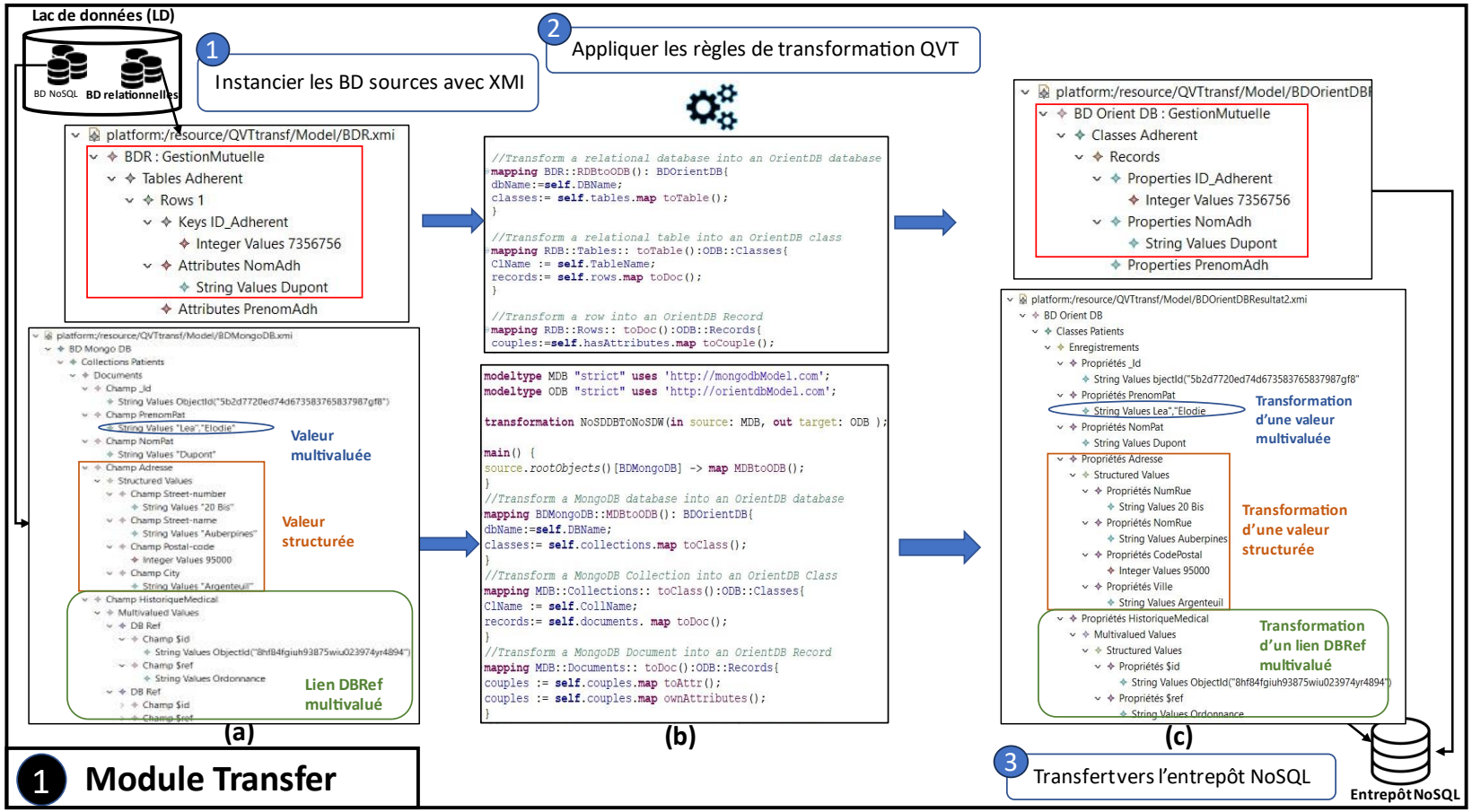
[46] R. Hai, C. Quix, et D. Wang, « Relaxed functional dependency discovery in heterogeneous data lakes », in *Conceptual Modeling: 38th International Conference, ER 2019, Salvador, Brazil, November 4–7, 2019, Proceedings 38*, Springer, 2019, p. 225-239.
[47] J. W. Ansari, « Semantic profiling in data lake », *RWTH Aachen University, Germany*, vol. 81, 2018.
[48] R. G. G. Cattell, D. K. Barry, et M. Berler, *The object data standard: ODMG 3.0*. Morgan Kaufmann, 2000.
[49] P. Asanka, « ETL FRAMEWORK DESIGN FOR NOSQL DATABASES IN DATA WAREHOUSING », *INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS*, vol. 3, déc. 2015.
[50] R. Yangui, A. Nabli, et F. Gargouri, « ETL based framework for NoSQL warehousing », in *Information Systems: 14th European, Mediterranean, and Middle Eastern Conference, EMCIS 2017, Coimbra, Portugal, September 7-8, 2017, Proceedings 14*, Springer, 2017, p. 40-53.
[51] L. Stanescu, M. Brezovan, et D. D. Burdescu, « Automatic mapping of MySQL databases to NoSQL MongoDB », in *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE, 2016, p. 837-840.
[52] A. A. Mahmood, « Automated algorithm for data migration from relational to NoSQL databases », *ALNAHRAIN JOURNAL FOR ENGINEERING SCIENCES*, vol. 21, n° 1, p. 60-65, 2018.
[53] M. Hanine, A. Bendarag, et O. Boutkhoum, « Data migration methodology from relational to NoSQL databases », *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 9, n° 12, p. 2566-2570, 2015.
[54] E. M. Kuszera, L. M. Peres, et M. D. D. Fabro, « Toward RDB to NoSQL: transforming data with metamorfose framework », in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, p. 456-463.
[55] R. Arora et R. R. Aggarwal, « An algorithm for transformation of data from MySQL to NoSQL (MongoDB) », *International Journal of Advanced Studies in Computer Science and Engineering*, vol. 2, n° 1, p. 6-12, 2013.
[56] D. S. Faster, « Pentaho Data Integration ». 2014.
[57] G. Liyanaarachchi, L. Kasun, M. Nimesha, K. Lahiru, et A. Karunasena, « MigDB-relational to NoSQL mapper », in *2016 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*, IEEE, 2016, p. 1-6.
[58] L. Rocha, F. Vale, E. Cirilo, D. Barbosa, et F. Mourão, « A framework for migrating relational datasets to NoSQL », *Procedia Computer Science*, vol. 51, p. 2593-2602, 2015.
[59] M. C. Freitas, D. Y. Souza, A. C. Salgado, et others, « Conceptual Mappings to Convert Relational into NoSQL Databases. », in *ICEIS (1)*, 2016, p. 174-181.
[60] A. Erraji, A. Maizate, et M. Ouzzif, « An integral approach for complete migration from a relational database to MongoDB », *Journal of the Nigerian Society of Physical Sciences*, p. 1089-1089, 2023.

[61] S. Ramzan, I. S. Bajwa, B. Ramzan, et W. Anwar, « Intelligent data engineering for migration to NoSQL based secure environments », *IEEE Access*, vol. 7, p. 69042-69057, 2019.
[62] F. Z. Belkadi et R. Esbai, « A Model-Driven Engineering: From Relational Database to Document-oriented Database in Big Data Context. », in *ICSOFT*, 2021, p. 653-659.
[63] Y. S. Wijaya et A. A. Arman, « A framework for data migration between different datastore of NoSQL database », in *2018 International Conference on ICT for Smart Society (ICISS)*, IEEE, 2018, p. 1-6.
[64] H. Dabbèchi, N. Z. Haddar, H. Elghazel, et K. Haddar, « Social media data integration: From data lake to nosql data warehouse », in *International Conference on Intelligent Systems Design and Applications*, Springer, 2020, p. 701-710.
[65] S. Wang, G. Pasi, L. Hu, et L. Cao, « The Era of Intelligent Recommendation: Editorial on Intelligent Recommendation with Advanced AI and Learning. », *IEEE Intell. Syst.*, vol. 35, n° 5, p. 3-6, 2020.
[66] Z. Fayyaz, M. Ebrahimian, D. Nawara, A. Ibrahim, et R. Kashef, « Recommendation systems: Algorithms, challenges, metrics, and business opportunities », *applied sciences*, vol. 10, n° 21, p. 7748, 2020.
[67] B. Alhijawi et Y. Kilani, « A collaborative filtering recommender system using genetic algorithm », *Information Processing & Management*, vol. 57, n° 6, p. 102310, 2020.
[68] U. Javed, K. Shaukat, I. A. Hameed, F. Iqbal, T. M. Alam, et S. Luo, « A review of content-based and context-based recommendation systems », *International Journal of Emerging Technologies in Learning (iJET)*, vol. 16, n° 3, p. 274-306, 2021.
[69] G. Adomavicius et A. Tuzhilin, « Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions », *IEEE transactions on knowledge and data engineering*, vol. 17, n° 6, p. 734-749, 2005.
[70] R. Burke, A. Felfernig, et M. H. Göker, « Recommender systems: An overview », *Ai Magazine*, vol. 32, n° 3, p. 13-18, 2011.
[71] E. B. Ahmed, A. Nabli, et F. Gargouri, « A survey of user-centric data warehouses: from personalization to recommendation », *arXiv preprint arXiv:1107.1779*, 2011.
[72] P. Marcel et E. Negre, « A survey of query recommendation techniques for data warehouse exploration. », in *EDA*, 2011, p. 119-134.
[73] M. Eirinaki et S. Patel, « QueRIE reloaded: Using matrix factorization to improve database query recommendations », in *2015 IEEE International Conference on Big Data (Big Data)*, IEEE, 2015, p. 1500-1508.
[74] M. Drosou et E. Pitoura, « Ymaldb: exploring relational databases via result-driven recommendations », *The VLDB Journal*, vol. 22, n° 6, p. 849-874, 2013.
[75] N. Khoussainova, Y. Kwon, M. Balazinska, et D. Suciu, « SnipSuggest: Context-aware autocompletion for SQL », *Proceedings of the VLDB Endowment*, vol. 4, n° 1, p. 22-33, 2010.

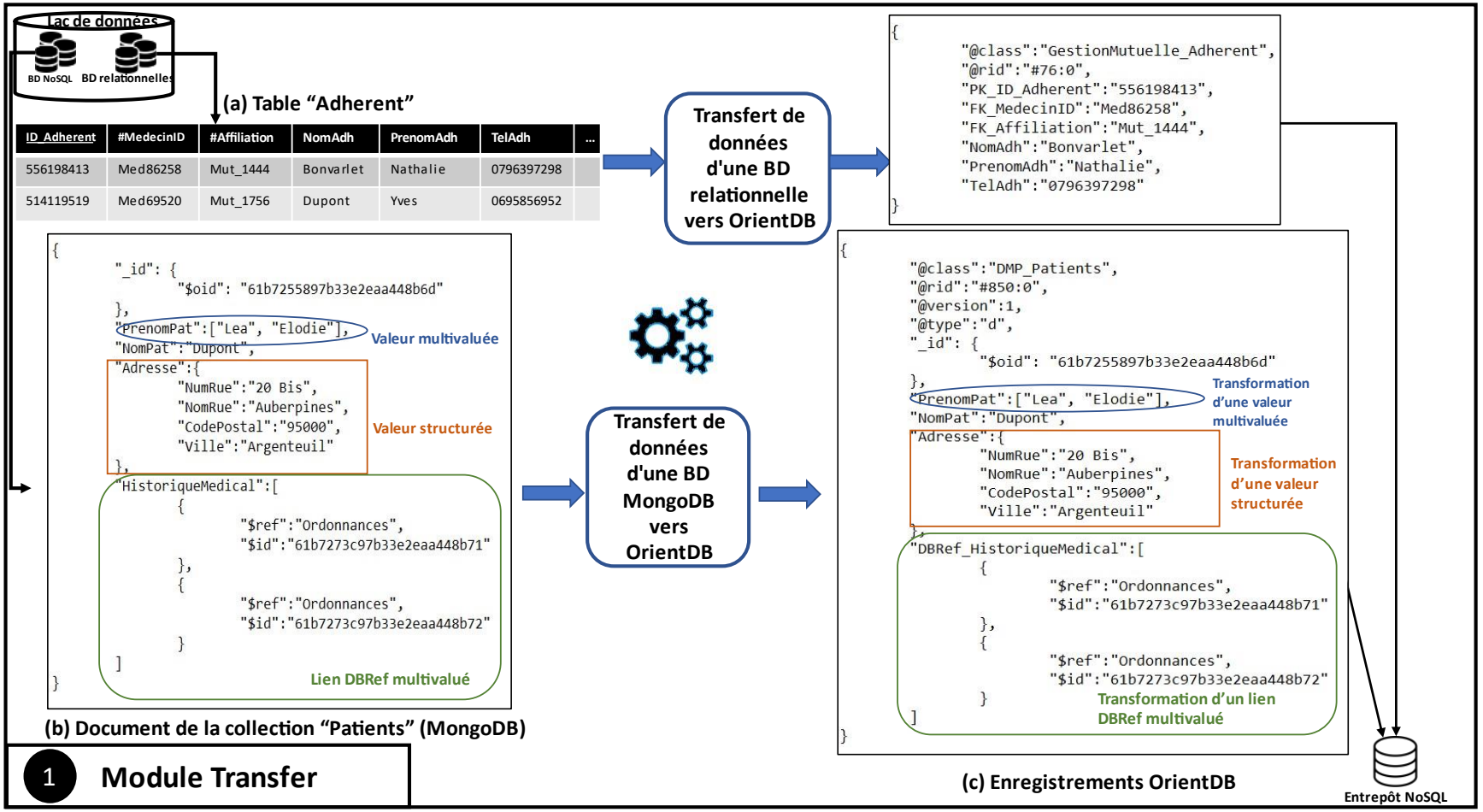
[76] S. Tahery et S. Farzi, « Customized query auto-completion and suggestion—A review », *Information Systems*, vol. 87, p. 101415, 2020.
[77] L. Ahmedi et D. Shabani, « Search engine query recommendation-using SNA over query logs with user profiles », in *International Conference on Web Information Systems and Technologies*, SCITEPRESS, 2017, p. 370-375.
[78] K. Drushku, J. Aligon, N. Labroche, P. Marcel, et V. Peralta, « Interest-based recommendations for business intelligence users », *Information Systems*, vol. 86, p. 79-93, 2019.
[79] A. Giacometti, P. Marcel, E. Negre, et A. Soulet, « Query Recommendations for OLAP Discovery-driven Analysis ».
[80] A. Giacometti, P. Marcel, et E. Negre, « A framework for recommending OLAP queries », in *Proceedings of the ACM 11th international workshop on Data warehousing and OLAP*, Napa Valley California USA: ACM, oct. 2008, p. 73-80.
[81] J. Bézivin et O. Gerbé, « Towards a precise definition of the OMG/MDA framework », in *Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001)*, IEEE, 2001, p. 273-280.
[82] E. Seidewitz, « What models mean », *IEEE software*, vol. 20, n° 5, p. 26-32, 2003.
[83] J. Hutchinson, J. Whittle, M. Rouncefield, et S. Kristoffersen, « Empirical assessment of MDE in industry », in *Proceedings of the 33rd international conference on software engineering*, 2011, p. 471-480.
[84] M. Belaunde *et al.*, « MDA Guide Version 1.0. 1 ». 2003.
[85] C. J. F. Candel, D. S. Ruiz, et J. J. García-Molina, « A Unified Metamodel for NoSQL and Relational Databases », *arXiv:2105.06494 [cs]*, mai 2021,
[86] A. Azqueta-Alzúaz, M. Patiño-Martinez, I. Brondino, et R. Jimenez-Peris, « Massive Data Load on Distributed Database Systems over HBase », in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2017, p. 776-779.
[87] M. T. Özsu et P. Valduriez, « Distributed and Parallel Database Systems », *ACM Comput. Surv.*, vol. 28, n° 1, p. 125-128, mars 1996.
[88] F. T. Machado, D. de Brum Saccol, E. Piveta, R. Padilha, et E. Ribeiro, « A Text Similarity-based Process for Extracting JSON Conceptual Schemas. », in *ICEIS (1)*, 2021, p. 264-271.
[89] L. F. Donnelly, R. Grzeszczuk, et C. V. Guimaraes, « Use of Natural Language Processing (NLP) in Evaluation of Radiology Reports: An Update on Applications and Technology Advances », *Seminars in Ultrasound, CT and MRI*, vol. 43, n° 2, p. 176-181, avr. 2022.
[90] F. Budinsky, *Eclipse modeling framework: a developer's guide*. Addison-Wesley Professional, 2004.
[91] B. Combemale, « Ingénierie Dirigée par les Modèles (IDM)–État de l'art », 2008.
[92] X. Blanc et O. Salvatori, *MDA en action: Ingénierie logicielle guidée par les modèles*. Editions Eyrolles, 2011.
[93] C. Manning, P. Raghavan, et H. Schuetze, « Introduction to Information Retrieval », 2009.

[94] S. Bird, E. Klein, et E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

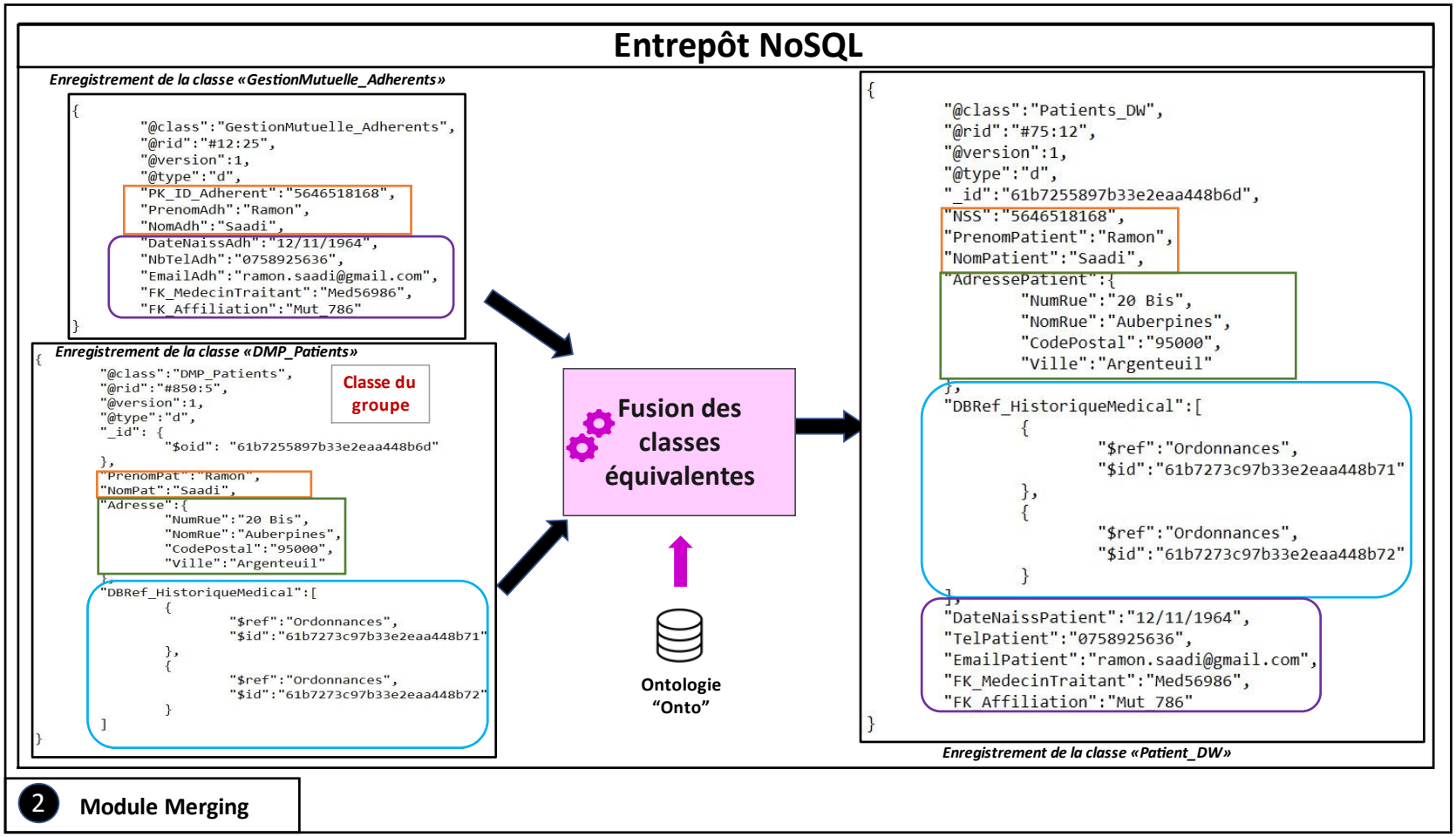
ANNEXES AGRANDISSEMENT DE FIGURES



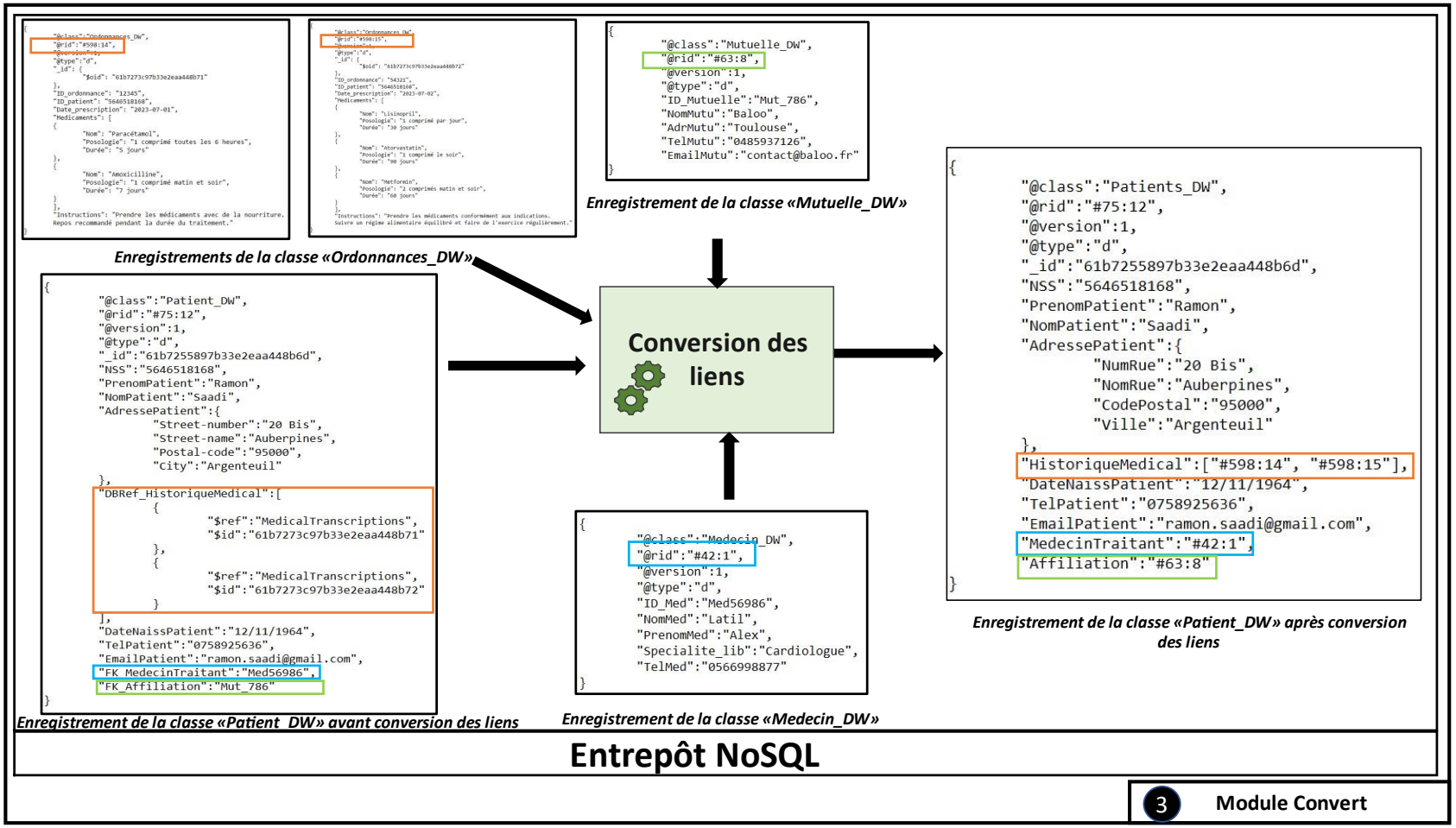
Annexe A - Figure 5.12 – page 110 – Étapes du transfert des données relationnelles et MongoDB selon l'architecture MDA



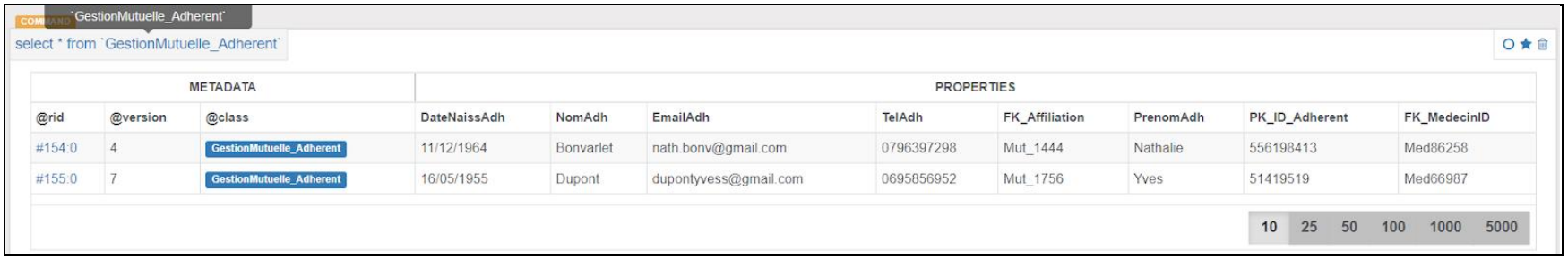
Annexe B - Figure 5.13– page 110 – Flux de données et traitements dans le module *Transfer*



Annexe C - Figure 5.15 – page 113 – Flux de données et traitements dans le module *Merging*



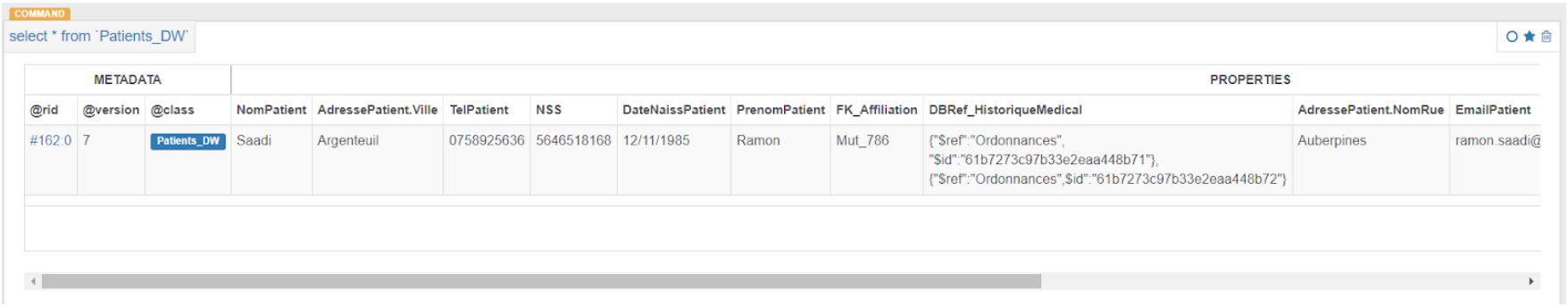
Annexe D - Figure 5.17 – page 114 – Conversion des liens relationnels et DBRef dans le module *Convert*



METADATA			PROPERTIES							
@rid	@version	@class	DateNaissAdh	NomAdh	EmailAdh	TelAdh	FK_Affiliation	PrenomAdh	PK_ID_Adherent	FK_MedecinID
#154:0	4	GestionMutuelle_Adherent	11/12/1964	Bonvarlet	nath.bonv@gmail.com	0796397298	Mut_1444	Nathalie	556198413	Med86258
#155:0	7	GestionMutuelle_Adherent	16/05/1955	Dupont	dupontyvess@gmail.com	0695856952	Mut_1756	Yves	51419519	Med66987

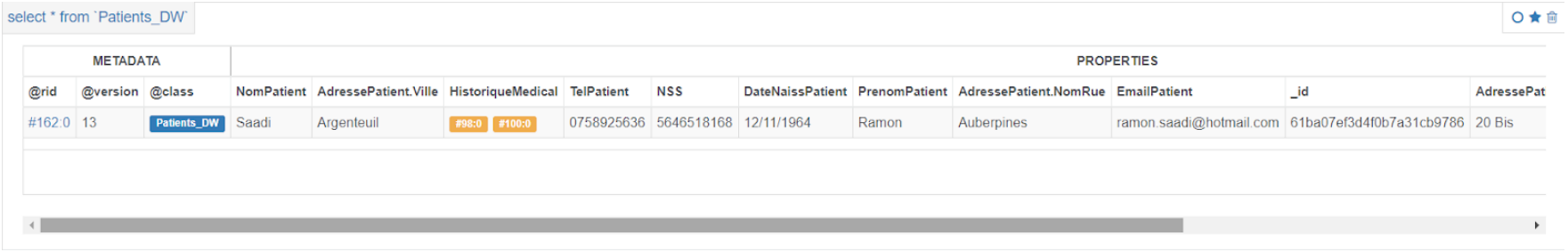
10 25 50 100 1000 5000

Annexe E - Figure 5.14 – page 111 – Résultat du transfert après exécution du module *Transfer*



METADATA			PROPERTIES									
@rid	@version	@class	NomPatient	AdressePatient.Ville	TelPatient	NSS	DateNaissPatient	PrenomPatient	FK_Affiliation	DBRef_HistoriqueMedical	AdressePatient.NomRue	EmailPatient
#162:0	7	Patients_DW	Saadi	Argenteuil	0758925636	5646518168	12/11/1985	Ramon	Mut_786	{"$ref":"Ordonnances", "$id":"61b7273c97b33e2eaa448b71"}, {"$ref":"Ordonnances",$id":"61b7273c97b33e2eaa448b72"}	Auberpines	ramon.saadi@

Annexe F - Figure 5.16 – page 113 – Résultat de la fusion après exécution du module *Merging*



Annexe G - Figure 5.18 – page 114 – Résultat de conversion des liens après exécution du module *Convert*