



N° d'ordre : 100/2021

Centre des Etudes Doctorales : Sciences et Techniques et Sciences Médicales
Formation Doctorale: **Sciences de l'Ingénieur, Sciences Physiques, Mathématiques et Informatique (SISPMI)**

Discipline : **Sciences et Techniques d'Information et de Communication.**

Spécialité: **Informatique**

Laboratoire : **Systèmes Intelligents et Applications**

Thèse pour l'obtention de DOCTORAT

La construction d'une base de connaissance à partir d'une Base de données relationnelles

Préparée par : Mr Bilal Ben Mahria

Soutenue le 24 Décembre 2021 devant le jury :

Nom et Prénom	Grade	Etablissement	Qualité
Pr. Ahlame Begdouri	PES	Faculté des Sciences et Techniques Fès	Président
Pr. Mohammed Bahaj	PES	Faculté des Sciences et Techniques Settat	Rapporteur
Pr. El moukhtar Zemmouri	PH	Ecole Nationale Supérieure des Arts et Métiers Meknès	Rapporteur
Pr. Arsalane Zarghili	PES	Faculté des Sciences et Techniques Fès	Rapporteur
Pr. Said Najah	PH	Faculté des Sciences et Techniques Fès	Examineur
Pr. Ilham Chaker	PH	Faculté des Sciences et Techniques Fès	Examineur
Pr. Azeddine Zahi	PES	Faculté des Sciences et Techniques Fès	Directeur de thèse

ACKNOWLEDGEMENTS

First and foremost, praises and thanks to Allah, the almighty, for His showers of blessings throughout my research work to complete the research successfully.

I would like to express my deep and sincere gratitude to my research supervisors: Dr. Azeddine Zahi and Dr. Ilham Chaker, for allowing me to do my research and for providing invaluable guidance through this research. Their dynamism, vision, sincerity and motivation have deeply inspired me. They have taught me the methodology to carry out the research and to present the research works as clearly as possible. It was a great privilege and honor to work and study under their guidance. I am extremely grateful for what they have offered me. I would also like to thank them for their friendship, empathy, and great sense of humor.

I am extremely grateful to my parents for their love, prayers, care and sacrifices in educating and preparing me for the future. My parents gave me the greatest gift not anyone could give...They believe in me.

Also I express my thanks to my sister, brothers, for their support and valuable prayers. I am very much thankful to my friends: Lamiae, Rachid, Imad, Najib, Mohcine for their understanding, prayers and continuing support to complete this dissertation.

Finally, I will send a message to me: Don't be happy, this is just beginning, you don't have to stop, just go on and after any reaching goal, start thinking in new goals. Do not follow where the path may lead. Go instead where there is no path and leave a trail.

Résumé de la thèse

Au cours des deux dernières décennies, l'intégration de données basées sur la sémantique est devenue l'un des plus grands défis en matière de données où les données sont publiées dans de nombreux formats différents (par exemple, HTML, CSV, JSON, feuille de calcul, données SQL, entre autres) et suivent plusieurs schémas. Par conséquent, l'utilisation de graphes de connaissances permet de résoudre les problèmes liés à l'intégration des données.

Le graphe de connaissances représente une collection de descriptions interconnectées d'entités (objets et événements du monde réel, ou concepts abstraits tels que des documents) où ces descriptions ont une sémantique formelle qui permet aux personnes et aux ordinateurs de les traiter de manière efficace et sans ambiguïté. En fait, la formation d'une sémantique formelle pour les entités est réalisée en utilisant des ontologies qui peuvent être considérées comme un schéma du graphe de connaissances. Les ontologies servent de contrat formel entre les développeurs du graphe de connaissances et ses utilisateurs, garantissant une compréhension commune des données et de leur signification.

Le processus de développement d'une ontologie est une activité d'ingénierie et il existe deux approches principales pour sa construction - soit à partir de zéro, soit en utilisant des méthodes d'apprentissage d'ontologie. La construction d'une ontologie à partir de zéro ou manuellement est une tâche très compliquée et coûteuse qui nécessite généralement une combinaison des connaissances des experts du domaine et des compétences des ingénieurs en ontologie. En raison de ces limitations, le terme "apprentissage d'ontologie" est apparu, qui capture une approche pour découvrir des connaissances ontologiques automatiquement ou semi-automatique à partir d'une source d'information structurée, non structurée ou semi-structurée.

Dans cette thèse, nous nous sommes concentrés sur une source d'information structurée, spécifiquement les bases de données relationnelles pour construire une ontologie pour plusieurs raisons. Premièrement, environ 70% des données sur le web sont stockées dans des bases de données relationnelles. Deuxièmement, les bases de données relationnelles présentent des modèles conceptuels complets. Troisièmement, elles fournissent une ressource d'information complète. Enfin, elles offrent l'une des meilleures techniques de stockage et de manipulation des données.

Cette thèse aborde le problème de la construction d'ontologie à partir de bases de données relationnelles en exploitant des règles de mapping déclaratives. Les contributions présentées dans cette thèse sont:

- Un cycle de vie complet pour la construction d'ontologie à partir de bases de données relationnelles basées sur les exigences du génie logiciel et du Big Data.
- Nous décrivons un processus de construction d'ontologie à partir d'une base de données relationnelle basée sur le cycle de vie prédéfini.

- Nous proposons un processus d'évaluation basé sur deux catégories : La vérification et la validation. La vérification permet de contrôler si l'ontologie a été construite correctement. Elle est divisée en deux sous-catégories : Vérification de la T-Box et Vérification de la A-Box. D'autre part, la validation de l'ontologie mesure la complexité de l'ontologie pour aider les ontologues à mieux comprendre, réutiliser, maintenir et intégrer les ontologies.
- Enfin, nous présentons une étude empirique pour répondre à la question de savoir comment sélectionner la stratégie la plus pertinente pour stocker les ontologies.

Les conclusions finales de cette thèse ont démontré le profond potentiel d'amélioration de l'efficacité et de l'efficience du processus de développement d'ontologies à partir de bases de données relationnelles. Enfin, les futures lignes de travail reflètent la nécessité de continuer à rechercher de nouvelles idées pour affiner le cycle de vie proposé afin d'aborder les questions suivantes : Nettoyage des données, alignement des ontologies, intégration des ontologies. De plus, le besoin de construire un cadre pour l'évaluation des ontologies devient une nécessité.

Mots clés : Ontologie, base de données relationnelle, TBox, ABox, ontologie cycle de vie, Evaluation d'ontologie, dépôt sémantique

Abstract

Over the last two decades, semantic-based data integration has become one of the biggest data challenges where the data is published in many different formats (e.g., HTML, CSV, JSON, spreadsheet, SQL data, among others) and follows several schemes. Therefore, the use of knowledge graphs provides support to address issues related to data integration.

The knowledge graph represents a collection of interlinked descriptions of entities (real-world objects and events, or abstract concepts such as documents) where these descriptions have formal semantics that allow both people and computers to process them in an efficient and unambiguous manner. In fact, forming a formal semantic for entities is performed by using ontologies that can be seen as a schema of the knowledge graph. The ontologies serve as a formal contract between the developers of the knowledge graph and its users that ensures a shared understanding of the data and its meaning.

The process of developing an ontology is an engineering activity and there are two main approaches for its construction -either from scratch, or by using ontology learning methods. Constructing ontology from scratch or manually is a very complicated and expensive task that usually requires a combination of the knowledge of domain experts and skills of ontology engineers. Due to these limitations, the term “ontology learning” has appeared, which captures an approach to discover ontological knowledge automatically or semi-automatically from a structured, unstructured, or semi-structured source of information.

In this thesis, we focused on a structured source of information, specifically relational databases to build ontology for several reasons. Firstly, around 70% of data on the web is stored in relational databases. Secondly, relational databases present full conceptual models. Thirdly, they provide a full information resource. Finally, they offer one of the best techniques for storing and manipulating data.

This thesis addresses the problem of construction ontology from relational databases exploiting declarative mapping rules. The contributions presented in this dissertation are:

- A complete life cycle for constructing ontology from relational databases based on the software engineering and Big Data requirements.
- We describe a process for building ontology from a Relational database based on the predefined life cycle.
- We suggest an evaluation process based on two categories: Verification and Validation. The verification to check if the ontology has been constructed correctly. It is divided into two sub-categories: Verification of the T-Box and Verification of the A-Box. On the other hand, the ontology validation measures the complexity of the ontology to help ontologists better understand, reuse, maintain and integrate ontologies.
- Finally, we introduce an empirical study to address the issue of how to select the most relevant strategy for storing the ontologies.

The final conclusions of this thesis have demonstrated the profound potential of improving the effectiveness and efficiency of the process of developing ontologies from relational databases. Finally, the future lines of work reflect the need to continue researching new ideas to

refine the proposed lifecycle to address the following issues: Data Cleaning, ontology alignment, ontology Integration. Furthermore, the need to build a framework for evaluating ontologies becomes a necessity.

Keywords : Ontology, relational database, TBox, ABox, triplestore, ontology lifecycle, Ontology Evaluation.

Contents

ACKNOWLEDGEMENTS	2
Résumé de la thèse	3
Abstract	5
LIST OF FIGURES.....	11
LIST OF TABLES	12
LIST OF ABBREVIATIONS	12
Introduction Générale.....	14
Chapter1. Un nouveau approche pour convertir une base de donnée relationnelles en ontologie 19	
1. Transformer une base de données relationnelles en Ontologie (LOFRDB): Cycle de vie	19
1.1. Discovery	20
1.2. Preparation.....	20
1.3. Development.....	21
1.4. Evaluation.....	21
2. Cycle de vie: Implementation	21
2.1. BDR Exploration	22
2.2. La construction TBox (Conceptual ontology).....	23
2.2.1. Transformation de contrainte Check	25
2.2.2. Transformation de contrainte default	25
2.2.3. Amélioration de la relation d'héritage	25
2.2.4. Algorithme pour construire le TBox	26
2.3. Algorithme pour construire le ABOX	27
2.3.1. Extraction de Meta-Données (DMEE).....	28
1.1.1. Générateur de BDR (MGE).....	28
1. Résultats et Discussion.....	29

1.1. La phase Discovery	29
1.2. BDR Exploration	31
1.3. De BDR à Ontologie: Sakila Database	33
Conclusion.....	35
Chapiter2. L’Evaluation de L’ontologie: Vérification Et Validation.....	37
1. Les approches de l’évaluation	37
1.1. Gold Standard	37
1.2. Data Driven.....	38
1.3. Task-Based	38
1.4. User Based.....	39
1.5. Metric Based.....	39
2. Les categories de l’évaluation	39
2.1. La verifcation de l’ontologie	39
2.2. La Validation de l’ontologie.....	39
3. La vérification de l'ontologie générée à partir de la base de données relationnelle	40
3.1. L’ évaluation de TBox	40
3.1.1. Attribute Richness (AR).....	40
3.1.2. Inheritance Richness (IR).....	40
3.1.3. Relationship Richness (RR)	41
3.2. La validation A-Box	41
3.2.1. Class Richness (CR).....	41
3.2.2. Average Population (AP)	42
4. Métriques de complexité avancées de l'ontologie	42
4.1. Métriques de complexité avancées de l'ontologie	42
4.1.1. SOV	42
4.1.2. Tree impurity (TIP)	42
4.1.3. Average number of paths per concept.....	43
4.1.4. The average path length	43

4.1.5. Coupling.....	43
5. Expériences d'évaluation d'ontologie	44
5.1. La verifcation de l'ontologie	44
5.1.1. L'évaluation TBox	45
1.1.1. L'évaluation ABox.....	46
1.2. La validation de l'ontology.....	46
1.2.1. Datasets	46
1.2.2. Result and Discussions.....	48
Conclusion.....	52
Chapter3. Les systèmes de stockage RDF	54
1. Importance du référentiel sémantique	54
1.1. Definition: Référentiel Sémantique	54
1.2. Pourquoi Référentiel Sémantique	55
2. La classification des systems de stockage RDF	56
2.1. Systèmes Native	56
2.2. Systèmes Non-Native	57
3. Implémentation proposées.....	59
3.1. RDF Subject-Predicate-Object method (RDFSPO)	60
3.2. RDF Property Clustering method (RDFPC).....	60
3.3. RDF Vertical Partitioning method (RDFVP)	61
4. Configuration des expériences	62
4.1. Description des systems RDF existants.....	63
4.2. Implémentation des requites.....	64
5. Résultats expérimentaux et discussion	65
5.1. L'évaluation des systems proposés.....	66
5.1.1. Mesure du temps de chargement	66
5.1.2. Temps de réponse des requêtes	66
5.2. L'évaluation des systèmes RDF existants	67

5.2.1. Mesure du temps de chargement	67
5.2.2. Le temps de réponses des requêtes.....	68
5.3. Comparaison des systèmes proposées contre systèmes existants	68
5.3.1. Le temps du chargement	69
5.3.2. Temps de réponses des requites	70
5.4. Discussion.....	71
Conclusion.....	74
Chapter4. Conclusions et Perspectives.....	75
1. Conclusions	75
2. Orientations futures de la recherche	77
List of Publications.....	78
References	79

LIST OF FIGURES

Figure 8: Life Cycle of learning ontologies from relational database.	19
Figure 9: The method of building the ontology from RDB.....	22
Figure 10: Check constraint for Data Range Restriction.....	25
Figure 11: Default Value constraint transformation.....	25
Figure 12: Improvement of the Inheritance relationship Example.....	26
Figure 13: Sakila Database EER Diagram	34
Figure 14: Sakila Ontology Constructed from sakila database	35
Figure 15: Discriminative effect of the knowledge coverage.....	45
Figure 16: The primitive metrics of the studied CH ontologies.....	48
Figure 17: The Size of Vocabulary of the studied ontologies.	49
Figure 18: the average path length and number of Paths per concepts.	49
Figure 19: The Tree Impurity of the studied ontologies.....	50
Figure 20: The coupling measure of the studied ontologies.....	51
Figure 21: Classification of the RDF storage systems	56
Figure 22 Load Time. Figure (a) shows the load time of the proposed methods. Figure(b) shows the existing methods:	66
Figure 23: load time of the proposed methods against the existing ones.	68
Figure 24: Comparison of the proposed methods against the existing ones in terms of scalability.....	69
Figure 25: Query response time comparison between the proposed methods and existing ones.....	70
Figure 26: Computational performance comparison of the proposed RDFVP, RDFSPO and RDFPC against the existing methods with Nemenyi Test.	73

LIST OF TABLES

Table 6: Summary of patterns to calculate NS.....	23
Table 7: The applied rules for generating Conceptual Ontology (TBox).....	24
Table 8: A list of metadata extracted from RDB.....	29
Table 9: The list of questions the ontologist must answer before start building ontology.....	30
Table 10: A list of Competency Questions.....	31
Table 11: The set of patterns.	32
Table 12: The NS and the total number of semantics for each database.....	33
Table 13: The basic metrics of the GoodRelations versus the resulting ontology.	44
Table 14: The list of metrics for evaluating the resulting ontology.	45
Table 15: the studied CH ontologies.	47
Table 16: Statement table approach.	58
Table 17: Clustered property table (CP).....	59
Table 18: Property Class approach (PC).	59
Table 19: Vertical partitioning approach (VP).....	59
Table 20: The basic Statistics of Datasets	63
Table 21: Characteristics of proposed RDF storage systems	64
Table 22: The tested queries implemented in this experimental study.....	64
Table 23: Characteristics of tested queries. SQ= Start Query, TQ=Tree Query and CQ= Chain Query. ...	65
Table 24: Comparison of Load Time (ms) metric of the proposed and existing methods on 17 data sets. The ranks in the parentheses (·) are used in the computation of the Friedman test.	72
Table 25 Comparison of Query Response Time (ms) metric of the proposed and existing methods on 12 queries. The ranks in the parentheses (·) are used in the computation of the Friedman test.	72

LIST OF ABBREVIATIONS

Abbreviation	Explanation
ML	Machine Learning
KG	Knowledge Graph
RDB	Relational Database
ETL	Extract, Transform, Load

A-Box	Assertional Box
T-Box	Terminological Box
CH	Cultural Heritage
EKG	Enterprise Knowledge Graph
OWL	Ontology Web Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SPO	Subject-Predicate-Object
URL	Uniform Resource Locator
BGP	Basic Graph Pattern
NOR	Non-Ontological Resource
R2RML	Relational Database to Resource Description Framework
RML	RDF Mapping Language
CQ	Competency Question
AP	Average Population
CR	Class Richness
IR	Inheritance Richness
RR	Relationship Richness
AR	Attribute Richness
TIP	Tree Impurity
SOV	Size Of Vocabulary
PT	Property Table
CP	Clustered Property Table
VP	Vertical Partitioning
SQ	Start Query
TQ	Tree Query
CQ	Chain Query

Introduction Générale

Le terme "Knowledge Graph" a été inventé par Google en 2012, en référence à l'utilisation de connaissances sémantiques dans la recherche sur le Web, qui permet aux utilisateurs de rechercher des concepts telles que des personnes ou des lieux, plutôt que le simple mécanisme de correspondance des chaînes de caractères que nous utilisons dans les documents Web ("Things, not strings"). Inspirés par le succès de Google, les graphes de connaissances gagnent du terrain dans les principales entreprises d'information du monde. Plus précisément, les graphes de connaissances (KG) sont reconnus par de nombreuses industries comme une approche efficace de la gouvernance des données, de la gestion des métadonnées et de l'enrichissement des données, et sont de plus en plus utilisés comme technologie d'intégration des données. Par conséquent, une promesse centrale est que les données hétérogènes, c'est-à-dire les données provenant de sources de données différentes (Non structurée, semi-structurées et structurées), peuvent être harmonisées et connectées afin que la qualité des données qui en résulte puisse être utilisée pour des tâches ultérieures, telles que les tâches d'apprentissage automatique (Dessi et al. 2021).

Le graphe de connaissances (KG) représente une collection de descriptions interconnectées d'entités (objets et événements du monde réel, ou concepts abstraits tels que des documents) où les descriptions ont une sémantique formelle (à l'aide d'ontologies) qui permet aux personnes et aux ordinateurs de les traiter de manière efficace et sans ambiguïté ; ces entités contribuent les unes aux autres, formant un réseau, où chaque entité fournit un contexte pour leur interprétation. Par conséquent, les ontologies représentent l'épine dorsale de la sémantique formelle d'un graphe de connaissances. Elles peuvent être considérées comme le schéma du graphe de connaissances. Elles servent de contrat formel entre les développeurs du graphe de connaissances et ses utilisateurs concernant la signification des données qu'il contient. Un utilisateur peut être un autre être humain ou une application logicielle qui souhaite interpréter les données de manière fiable et précise. Les ontologies garantissent une compréhension commune des données et de leurs significations (Pan et al. 2017a).

Les éléments clés du développement d'un graphe de connaissances sont la compréhension des principales phases et tâches requises au cours du processus de développement. En bref, il existe trois couches principales liées à l'utilisation des graphes de connaissances : la couche d'acquisition des connaissances, la couche de stockage des connaissances et la couche de l'exploitation. Tout

d'abord, la couche d'acquisition des connaissances décrit le processus d'extraction des informations de différentes sources, leur structuration et la création de connaissances utiles. Les informations peuvent être représentées différemment selon les systèmes, comme des documents textuels, des pages Web et des bases de données. Ensuite, une fois la phase d'acquisition des données identifiée et terminée, l'étape suivante consiste à trouver comment stocker ces données (Pan et al. 2017b). Enfin, la phase d'exploitation des connaissances encodées dans le graphe de connaissances a commencé afin d'améliorer l'efficacité de vos données et de répondre à votre besoin spécifique (Pan et al. 2017b).

D'une manière générale, l'acquisition de connaissances est la couche où nous devons développer l'ontologie qui représente l'élément central du graphe de connaissances. En outre, le processus de développement d'une ontologie est une activité d'ingénierie et il existe deux approches principales pour sa construction - soit à partir de zéro, soit en utilisant des méthodes d'apprentissage d'ontologie (Ontology Learning). La construction d'une ontologie à partir de zéro ou manuellement (Grüninger and Fox 1995; Uschold and King 1995; Fernández-López et al. 1997; Noy and McGuinness 2001; Sure et al. 2004; Al-Arfaj and Al-Salman 2015) est une tâche très compliquée et coûteuse qui nécessite généralement une combinaison des connaissances des experts du domaine et des compétences des ingénieurs en ontologie. En raison de ces limitations, le terme "apprentissage d'ontologie" est apparu, qui capture une approche pour découvrir des connaissances ontologiques automatiquement ou semi-automatiquement à partir de diverses ressources telles que les bases de données relationnelles, les documents texte, etc. L'approche apprentissage d'ontologie peut résoudre les problèmes d'acquisition de connaissances et facilite grandement la construction d'ontologies par rapport à la méthode manuelle.

Dans cette thèse, nous nous sommes concentrés sur les bases de données relationnelles comme source d'information pour construire une ontologie pour plusieurs raisons. Premièrement, environ 70% des données sur le web sont stockées dans des bases de données relationnelles. Deuxièmement, les bases de données relationnelles présentent des modèles conceptuels complets. Troisièmement, elles fournissent une ressource d'information complète. Enfin, elles offrent l'une des meilleures techniques de stockage et de manipulation des données. Cependant, les bases de données relationnelles souffrent de l'absence de signification sémantique, ce qui entrave la capacité à réaliser l'interopérabilité entre les systèmes d'information.

Pour construire une ontologie à partir d'une base de données relationnelle, il y a trois étapes principales : Le schéma de mappage, la migration des données et l'accessibilité des données. Tout d'abord, le schéma de mappage comprend deux étapes principales : L'extraction des métadonnées des modèles de base de données (conceptuels, logiques ou physiques) et la génération de règles de mappage de chaque composant extrait dans la phase de métadonnées, puis leur conversion dans leur composant correspondant selon le langage de l'ontologie (Spanos et al. 2012). Il existe trois types de mappage : Le mappage automatique, le mappage semi-automatique et le mappage manuel. L'approche de mappage automatique vise à transformer une base de données relationnelle (RDB) en ontologie de manière simple, sans interaction de l'utilisateur dans le processus de mappage. L'approche de mappage semi-automatique a pris place pour améliorer un mappage de l'ontologie

locale en utilisant les classes et les propriétés des ontologies existantes qui sont déjà développées et publiées sur le web. Les approches de mappage manuel proposent un processus itératif dans lequel un expert du domaine valide ou rejette les mappages proposés.

Deuxièmement, le schéma de mappage n'est pas suffisant pour créer une ontologie à partir de la base de données, car la base de données relationnelle comprend à la fois des informations sur le schéma et des données sur les instances, qui doivent également être traitées. Par conséquent, la façon dont les données d'instances sont transformées en instances ontologiques est appelée migration des données. Deux méthodes peuvent être appliquées pour réaliser la phase de migration des données : la transformation statique et la transformation dynamique. La transformation statique génère un graphe RDF à partir d'une instance de base de données relationnelle en utilisant le processus ETL: Extraire, transformer et charger. La transformation dynamique repose sur la synchronisation des données, ce qui est important si les données sont fréquemment mises à jour. Dans ce cas, les données résident dans la base de données relationnelle, et la manière d'accéder aux données comprend la réécriture de la requête sémantique en une requête SQL, qui est exécutée, et son résultat est converti en réponse à la requête sémantique(Spanos et al. 2012).

Troisièmement, indépendamment de la manière dont l'ontologie est construite, l'accessibilité des données définit la manière dont les données sont interrogées et extraites de la connaissance construite. Par conséquent, il est possible d'accéder au contenu de l'ontologie en utilisant un langage de requête d'ontologie ou des données liées(Po et al. 2020).

Dans cette thèse, nous avons abordé les deux premières étapes qui sont : Le mappage des schémas et la migration des données.

Problématique:

Malgré les progrès importants réalisés au cours des dernières années et le grand nombre d'approches proposées, nombreuses questions n'ont pas encore été suffisamment traitées.

1. Toutes des études existantes se concentrent uniquement sur la description du processus de génération de l'ontologie à partir de la BDR, alors qu'elles n'ont pas défini un cycle de vie pour décrire les scénarios les plus courants qui se produisent pendant la création de l'ontologie à partir de la BDR.
2. Tous les travaux existants se concentrent uniquement sur la génération A-Box ou de T-Box et ignorent le processus d'intégration entre ces deux composants.
3. La majorité des études se sont principalement concentrées sur le processus de construction d'ontologies à partir de bases de données relationnelles sans couvrir la sémantique maximale résidant dans la base de données telle que la contrainte par défaut et la transitivité.

4. Un autre défi auquel on peut être confronté pendant le processus d'apprentissage des ontologies est la disponibilité des ontologies pour différents domaines sur le web qui progressivement augmenté. Par conséquent, l'ontologie résultante des BDR doit être évaluée sous différents angles afin de déterminer sa qualité avant son utilisation ou sa réutilisation. Dans ce contexte, tous les travaux existants sur ce sujet n'ont pas pris en considération la mesure de la qualité de l'ontologie résultante.

5. Une fois le processus de construction d'une ontologie terminé, la question du choix d'une stratégie appropriée pour le stockage de l'ontologie est abordée.

Contributions

Sur la base de la problématique susmentionnée, les principales contributions de cette thèse sont :

- Nous proposons un premier cycle de vie pour la construction d'ontologies à partir de bases de données relationnelles, basé sur les exigences du génie logiciel et du Big Data ;
- Nous décrivons un processus de construction d'ontologie à partir d'une base de données relationnelle basée sur le cycle de vie prédéfini
- Nous suggérons un processus d'évaluation basé sur deux catégories : La vérification et la validation. La vérification permet de contrôler si l'ontologie a été construite correctement. Elle est divisée en deux sous-catégories qui sont : Vérification de la T-Box et Vérification de la A-Box. D'autre part, la validation de l'ontologie mesure la complexité de l'ontologie pour aider les ontologistes à mieux comprendre, réutiliser, maintenir et intégrer les ontologies.
- Enfin, nous présentons une étude empirique pour répondre à la question de savoir comment sélectionner la stratégie la plus pertinente pour stocker les ontologies.

Le reste de cette thèse est organisé comme suit :

Dans le premier chapitre, nous présenterons toutes les connaissances de base nécessaires pour étudier et comprendre le graph de connaissance.

Dans le deuxième chapitre, nous nous concentrons sur l'introduction d'un état de l'art de la construction d'ontologies à partir de bases de données relationnelles. Nous donnons un aperçu des deux principales approches pour construire une ontologie. Nous présentons également une méthode de classification pour construire une ontologie à partir d'une base de données relationnelle. En outre, nous présentons deux approches importantes pour réaliser la phase de migration des données et nous traitons les deux principales méthodes pour récupérer les données du graphe de connaissances construit. Enfin, nous résumons les principaux langages de mapping qui peuvent être utilisés dans le processus de conversion d'une base de données relationnelle en graphe de connaissances.

Dans le troisième chapitre, nous présentons et explorons un nouveau cycle de vie pour l'apprentissage d'ontologies à partir de bases de données relationnelles, basé sur les exigences du génie logiciel et du big data. En outre, nous présentons un nouveau processus de construction d'ontologie à partir de bases de données relationnelles basé sur le cycle de vie prédéfini.

Dans le quatrième chapitre, nous nous concentrons sur la tâche d'évaluation pour mesurer la qualité de l'ontologie générée à partir de la base de données relationnelle. En outre, nous décrivons les métriques des fonctionnalités avancées pour analyser la complexité de certaines ontologies existantes dans le domaine du patrimoine culturel (CH).

Dans le cinquième chapitre, nous fournissons les aspects de performance les plus appropriés pour sélectionner les stratégies de stockage RDF pertinentes, et analysons leurs avantages et inconvénients. Dans ce contexte, nous fournissons une étude empirique basée sur l'analyse statistique, y compris le test post hoc de Friedman et Nemenyi.

Dans le dernier chapitre, nous présentons des conclusions et exposons de nouvelles idées, qui sortent du cadre de cette thèse pour des travaux futurs. Les contributions sont mises en évidence, et les futures directions de recherche possibles sont élaborées.

Chapter1. Un nouveau approche pour convertir une base de donnée relationnelles en ontologie

Dans ce chapitre, nous : (1) Présentons un nouveau cycle de vie pour l'apprentissage des ontologies à partir des BDR basé sur les exigences de l'ingénierie logicielle ; (2) Décrivons une nouvelle méthode pour construire une ontologie à partir d'une base de données relationnelle basée sur le cycle de vie prédéfini ; (3) Ajoutons trois nouvelles sémantiques qui peuvent être extraites des BDR.

1. Transformer une base de données relationnelles en Ontologie (LOFRDB): Cycle de vie

Comme le montre la figure 8, le cycle de vie que nous proposons est composé de quatre phases : Découverte, Préparation, Développement, et Evaluation. Pour la plupart des phases du cycle de vie, le mouvement peut être soit en avant, soit en arrière. Cette représentation itérative est destinée à décrire plus fidèlement un projet réel (Services 2015), dans lequel certains aspects du projet avancent et peuvent revenir à des étapes antérieures à mesure que de nouvelles informations sont découvertes et que l'ontologie en apprend davantage sur le domaine d'intérêt (Calero et al. 2006).

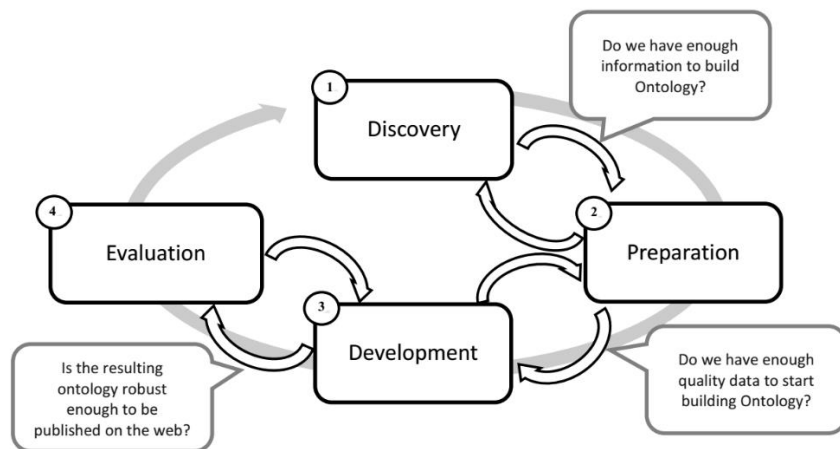


Figure 1: Life Cycle of learning ontologies from relational database.

1.1. Discovery

La première étape du cycle de vie comprend la découverte, au cours de laquelle l'ontologue doit apprendre et étudier le problème, développer le contexte et la compréhension, et se renseigner sur les sources de données nécessaires et disponibles pour le projet. En outre, l'ontologue et l'expert du domaine formulent des hypothèses initiales qui peuvent ensuite être testées avec des données.

Dans cette étape, l'ontologue doit clairement définir le domaine et la portée de l'ontologie en répondant aux questions suivantes (Noy and McGuinness 2001):

- Quel est le domaine que l'ontologie va couvrir ?
- Dans quel but allons-nous utiliser l'ontologie (application) ?
- Pour quels types de questions les informations contenues dans l'ontologie doivent-elles fournir des réponses ?
- Quelles sont les utilisations prévues de l'ontologie et qui sont les utilisateurs finaux (parties prenantes) ?
- Quelles sont les sources des BDR utilisées pour construire l'ontologie ?
- Est-il nécessaire d'interroger l'expert du domaine ?

Les réponses à ces questions peuvent changer au cours du développement de l'ontologie, mais à tout moment, elles permettent de limiter la portée du modèle. À ce stade, l'ontologue formule également des questions de compétence (CQ) auxquelles l'ontologie devrait pouvoir répondre et qui pourront être testées ultérieurement (Pan et al. 2017a). L'objectif des CQ est de vérifier si l'ontologie comprend suffisamment d'informations pour répondre à ces questions et si les réponses nécessitent un niveau de détail particulier ou la représentation d'un domaine particulier. Ces CQ ne sont qu'une ébauche et n'ont pas besoin d'être exhaustifs (Pan et al. 2017a).

Dans le cadre de la phase de découverte, l'ontologue doit évaluer les ressources disponibles pour soutenir le processus de développement de l'ontologie. Dans ce contexte, les ressources contiennent la technologie, les outils, les données et les personnes. En outre, l'ontologue peut supprimer ou ajouter des sources de données dans cette phase (Pan et al. 2017b).

1.2. Preparation

La deuxième phase de la LOFRDB concerne la préparation des données, qui comprend les étapes d'exploration et de prétraitement (conditionnement) préalable des données. L'exploration des données consiste à vérifier si les sources de données contiennent suffisamment de sémantique pour générer une ontologie en vérifiant si la BDR contient l'espace complet des relations et le maximum de combinaisons possibles des clés primaires et des clés étrangères (Sequeda et al. 2011). La deuxième sous-phase est le conditionnement des données, qui fait référence au processus de

nettoyage des données et de normalisation des ensembles de données. Nous pouvons considérer la normalisation de la BDR comme une partie de la phase de conditionnement des données.

1.3. Development

Le développement (la construction de l'ontologie) est abordé en deux phases : Le pré-développement et le post-développement. Le pré-développement commence par l'acquisition des données (ABox), qui consiste à extraire les instances de la base de données relationnelle (de Medeiros et al. 2015b), et à les représenter sous la forme de triplets RDF (Gutierrez et al. 2011). Après l'acquisition des données, l'acquisition du schéma (TBox) (Astrova et al. 2007) sera lancée afin de générer la définition et la signification des instances extraites. Par conséquent, il est nécessaire de construire un vocabulaire de ces termes pour simplifier la phase de développement. Le développement ne comprend pas seulement l'acquisition des données et des schémas, mais aussi la phase d'intégration de ces deux composants. Le post-développement comprend plusieurs autres tâches telles que l'alignement, la fusion et l'intégration, etc. (Calero et al. 2006)

1.4. Evaluation

Après avoir construit l'ontologie à partir de la BDR, des métriques pour évaluer l'ontologie résultante doivent être présentées (Lourdusamy and John 2018a). En général, le processus d'évaluation peut être défini comme le processus de décision sur la qualité de l'ontologie par rapport à des métriques particulières (Lourdusamy and John 2018a). À cette fin, deux dimensions orthogonales pour évaluer la qualité de l'ontologie résultante sont définies ; (i) la première dimension est l'évaluation T-Box ; (ii) la deuxième dimension est l'évaluation A-Box. L'évaluation de la T-Box postule la conception de la T-Box construite. Bien que nous ne puissions pas savoir avec certitude si la conception de la T-Box modélise correctement la connaissance du domaine, des mesures telles que la richesse et l'héritage indiquent la qualité de la T-Box créée. Les métriques les plus significatives dans cette catégorie sont décrites dans (Tartir et al. 2005).

2. Cycle de vie: Implementation

Du cycle de vie proposé, de nombreux processus ou modèles peuvent être extraits et cela dépend des besoins de l'ontologue et des objectifs du projet. Nous considérons que les données sont déjà nettoyées et conditionnées. De plus, l'ontologie résultante n'a besoin ni d'alignement ni de fusion avec d'autres ontologies.

Comme le montre la figure 9, après la phase de découverte, qui vise à identifier le domaine et la portée de l'ontologie ainsi qu'à jeter un premier coup d'œil aux sources de données, la phase suivante est la préparation des données. Dans cette phase, certaines caractéristiques sémantiques sont extraites et nous utilisons une nouvelle métrique pour choisir la RDB la plus pertinente. A

partir de cette dernière, nous générons la ABox et la TBox (Domingue et al. 2011a) puis après nous intégrons les deux composants pour obtenir l'ontologie finale. La dernière phase de notre processus est la validation de l'ontologie résultante qui consiste à évaluer les composants ABox et TBox en utilisant des métriques et une ontologie de référence, et enfin vérifier si l'ontologie résultante peut répondre aux questions de compétence (CQ) (Pan et al. 2017a). Si la validation (Fernández et al. 2009) échoue, cela signifie que l'ontologie résultante ne peut pas être publiée sur le web ou utilisée dans des applications. Dans ce cas, il faut revenir à la phase de découverte..

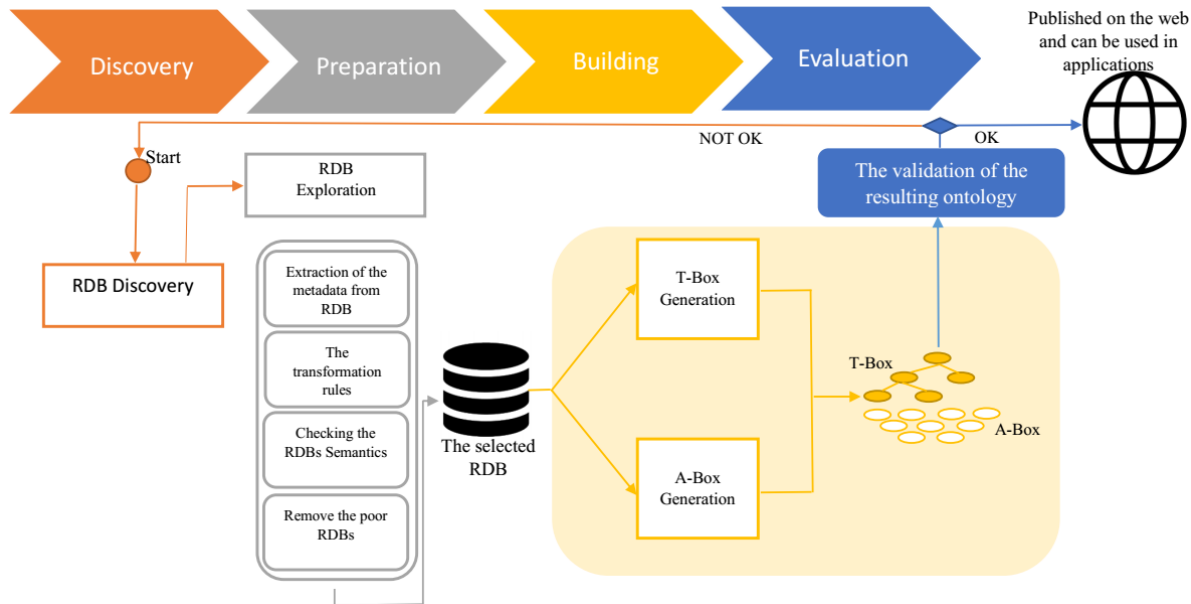


Figure 2: The method of building the ontology from RDB

2.1. BDR Exploration

La phase d'exploration consiste à vérifier si les bases de données relationnelles d'entrée contiennent l'espace complet de métadonnées et de caractéristiques sémantiques pour générer l'ontologie. Dans ce contexte, certaines informations peuvent être extraites des bases de données relationnelles d'entrée, comme le nombre de tables, de colonnes, de clés primaires, de clés étrangères et d'instances. D'autre part, nous considérons les caractéristiques sémantiques résumées dans le tableau 6 pour choisir la RDB la plus pertinente.

Patterns	Acronym
table without FKs	NTDFK
table with one FK	NTFK
table with more than 2 FKs	NTMTWOFK
Tables that contain exactly 2 foreign keys with presence of independent attributes.	NTEXTWOFK
attributes that are FK+ NULL + Not UNIQUE	NAFKNNU

attributes that are FK+ NOT NULL+ NOT UNIQUE	NAFKNNNU
attributes that are FK+ NOT NULL+ UNIQUE	NAFKNNU
attributes that are FK+ NOT NULL+UNIQUE+ NOT PK (FK is not equal to the PK)	NANNUNPK
attribute (neither PK nor FK)	NA
Attribute + NOT NULL	NAN
attribute NOT FK + UNIQUE	NANFKU
PK	NPK
Attribute with constraint with an integer greater than 0	
CHECK constraint with enumeration	NACHECK
CHECK constraint	
attribute with Default Value constraint	NADef
tables with tables share the same primary key	NTSAMEPK
FK is a reference to the same table	NUnaryRel
FK that is a reference to the same table, but it is accompanied by a trigger ON DELETE CASCADE	NTrRel

Table 1: Summary of patterns to calculate NS.

Dans ce contexte, nous proposons la métrique NS (Number of Semantics) qui représente le nombre de caractéristiques sémantiques de chaque RDB d'entrée. L'intervalle de cette métrique est de 0 à 17. Les valeurs proches de 0 reflètent une base de données relationnelle sémantiquement pauvre, tandis que les grandes valeurs, proches de 17, représentent une BDR riche. La métrique NS est calculée en donnant la valeur "1" à chaque caractéristique existant dans la BDR et "0" sinon :

$$NS = NTDFK + NTFK + NTMTWOFK + NTEXTWOFK + NAFKNNNU + NAFKNNU + NAFKNNU + NANNUNPK + NA + NAN + NANFKU + NPK + NACHECK + NADEF + NTSAMEPK + NUnaryRel + NTrRel.$$

L'exploration de la base de données relationnelle nécessite également une intervention humaine pour sélectionner la base de données relationnelle pertinente car une base de données qui a un nombre total élevé de sémantiques ne signifie pas qu'elle couvre toutes les sémantiques possibles (Spanos et al. 2012).

2.2. La construction TBox (Conceptual ontology)

La TBox introduit le vocabulaire d'un domaine d'application. Elle représente le référentiel qui contient les déclarations d'axiomes de concepts ou de rôles (Jimborean and Groza 2014). Pour générer la TBox à partir de RDB, nous utilisons certains patrons de transformation qui sont définis dans le tableau 7. De manière concise, les principales étapes de la génération d'une ontologie conceptuelle sont décrites dans l'algorithme 1. Dans cette étape, nous proposons 3 nouvelles règles de transformation qui permettent de transformer : la contrainte de vérification, la contrainte par défaut, et la contrainte d'amélioration de la relation d'héritage

Patterns	Kind of patterns	OWL corresponding element
Table patterns	Table without FK	OWL: class

	Table with one FK	
	Table with more than 2 FKs	
	Tables that contain exactly 2 foreign keys with presence of independent attributes.	
Binary Relationship table	Tables that contain exactly 2 foreign keys without presence of independent attributes.	We create two object properties (owl:objectProperty) The latter is an inverse of the former
Tables with one FK	Attributes that are FK+ NULL + Not UNIQUE	Object Property + Functional Property + Min Cardinality of the inverse property =1
	Attributes that are FK+ NOT NULL+ NOT UNIQUE	Object Property + Card=1 + Min Cardinality of the inverse property = 1
	Attributes that are FK+ NOT NULL+ UNIQUE	Object Property + Functional Property+ Functional Property for the inverse Property.
	Attributes that are FK+ NOT NULL+UNIQUE+ NOT PK (FK is not equal to the PK)	Object Property+ Functional Property+ Card=1 + Functional Property for the inverse Property.
Attributes	Attribute (neither PK nor FK)	DatatypeProperty
	Attribute + NOT NULL	DatatypeProperty + MinCardinality=1
	Attributes NOT FK + UNIQUE	DatatypeProperty + MaxCardinality=1
	Primary Key	MinCardinality+MaxCardinality=1 (Cardinality =1)
Check Constraint	Attribute with constraint with an integer greater than 0	xsd:positiveInteger
	CHECK with enumeration	xsd:positiveInteger
	CHECK constraint as Value Restriction	Xsd:minInclusive, Xsd:maxInclusive, Xsd:minExclusive, Xsd:maxExclusive
Default constraint	Attribute with Default Value	Owl:hasValue
Inheritance relationship	Two tables share the same primary key.	rdfs:subClassOf
Symmetric Relationship	FK is a reference to the same table	owl: SymmetricProperty
Transitive Relationship	FK is a reference to the same table, but now it is accompanied by a trigger ON DELETE CASCADE	OWL:TransitiveProperty.
Inheritance relationship improvement	The range of the foreign Key attribute	Owl:AllValuesFrom

Table 2: The applied rules for generating Conceptual Ontology (TBox).

2.2.1. Transformation de contrainte Check

Comme mentionné dans (Obrenović and Luković 2014), Check constraints are conditions that validates the data in a table. les contraintes Check sont des conditions qui valident les données d'une table. Dans cette section, nous proposons une règle pour transformer la contrainte CHECK en restriction de plage de données. Pour résoudre ce problème, nous avons utilisé les facettes de limites, qui sont : xsd:minInclusive, xsd:minExclusive, xsd:maxInclusive, et xsd:maxExclusive (Yu 2011b) (see Figure 10).

SQL DDL	OWL
<pre>CREATE TABLE Persons (Age int, CHECK (Age >= 18));</pre>	<pre><rdfs:Datatype rdf:about="&example;PersonAge"> <owl:withRestrictions rdf:parseType="Collection"> <rdf:Description> <xsd:minInclusive rdf:datatype="&xsd;integer">18</xsd:minInclusive> </rdf:Description> </owl:withRestrictions> </rdfs:Datatype></pre>

Figure 3: Check constraint for Data Range Restriction

2.2.2. Transformation de contrainte default .

La contrainte DEFAULT dans RDB (El Alami and Bahaj) est utilisée pour fournir une valeur par défaut pour une colonne. A cet égard, la contrainte owl:hasValue décrit une classe de tous les individus pour lesquels la propriété concernée a au moins une valeur sémantiquement égale à la valeur par défaut. Par conséquent, owl:hasValue indique quel que soit le nombre de valeurs d'une classe pour une propriété particulière, au moins l'une d'entre elles doit être égale à la valeur par défaut (Yu 2011b). La figure 11 illustre la transformation de la contrainte par défaut en OWL.

SQL DDL	OWL
<pre>CREATE TABLE Persons (City varchar(255) DEFAULT 'FEZ');</pre>	<pre><owl:Restriction> <owl:onProperty rdf:resource="#hasCity"/> <owl:hasValue rdf:datatype="&xsd;String">FEZ </owl:hasValue> </owl:Restriction></pre>

Figure 4: Default Value constraint transformation.

2.2.3. Amélioration de la relation d'héritage

Il est important de réaliser que dans OWL, les domaines et les plages ne doivent pas être considérés comme des contraintes à vérifier (Yu 2011b). Ils sont utilisés comme des "axiomes" dans le raisonnement. Par exemple, si la propriété hasProfessor a pour plage Professeur et pour domaine Étudiant, et que nous appliquons la propriété hasProfessor à Étudiant (les instances qui sont

membres de la classe Étudiant), cela n'entraînera généralement pas d'erreur. Sachant que Student et Professor sont des sous-classes de Person. Dans ce contexte, on en déduirait que les classes Student et Professor peuvent avoir des instances en commun. Plus précisément, on peut trouver que "Student hasProfessor Student". Par conséquent, nous utiliserons la contrainte owl : AllValuesFrom (Yu 2011b) pour éviter ce problème, comme le montre la figure 12.

SQL DDL	OWL
<pre>CREATE TABLE Person (Pid integer primary key, Name varchar(15) Not Null); CREATE TABLE Professor (id integer primary key, Title varchar(15), CONSTRAINT FK_Person FOREIGN KEY (id) REFERENCES Employee (pid));); CREATE TABLE Student (id integer primary key, Degree varchar(15), CONSTRAINT FK_Person FOREIGN KEY (id) REFERENCES Employee (pid)););</pre>	<pre><owl:ObjectProperty rdf:ID="hasProfessor"> <rdfs:domain rdf:resource="#Student"/> </owl:ObjectProperty> <owl:class rdf:about="#Student"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="hasProfessor"/> <owl:AllValuesFrom rdf:resource="#Professor"/> </owl:Restriction> </rdfs:subClassOf> </owl:Class></pre>

Figure 5: Improvement of the Inheritance relationship Example.

2.2.4. Algorithme pour construire le TBox

La TBox introduit la terminologie et le vocabulaire du domaine d'application. Elle représente le référentiel qui contient la déclaration des axiomes ou des rôles des concepts. Une approche naïve considérerait que la TBox correspond au schéma de la base de données relationnelle (Konstantinou and Spanos 2015). Dans cette phase, nous implémentons les règles qui sont identifiées dans le tableau 7. De manière concise, les principales étapes de la génération de la TBox sont décrites dans l'algorithme 1.

Algorithm 1: a Sketch of the T-Box Generation

```

Input: SQL DDL File
Output: OWL File
Var:
  manager: OWLOntologyManager
  arrTables: ArrayList<String>
  cp: ArrayList<TableComponent>

Begin
1: arrTables←getTables()
2: for each i<arrTables.size()start
3:   cp←getColumnsOfSpecificFieldTable(arrTables.get(i))
4:   if(isNotPrimaryKey(cp.getColName(), arrTables.get(j)) && cp.getNullable()=="YES"&& NotUnique)
5:     createOWLDataProperty();
6:   end if
7:   if(isNotPrimaryKey(cp.getColName(), arrTables.get(j)) && cp.getNullable()=="NO"&& NotUnique)
8:     createOWLDataPropertyWithMinCadrinalityEqualOne();
9:   end if
10:  if(isNotPrimaryKey(cp.getColName(), arrTables.get(j)) && cp.getNullable()=="YES"&& Unique)
```

```

11:         createOWLDataPropertyWithMaxCadrinalityEqualOne();
12:     end if
13:     if(isNotCompositePrimaryKey(arrTables.get(i)))
14:         if(isNotPrimaryKey(cp.getColName(), arrTables.get(j)))
15:             createOWLDataPropertyWithExactCadrinalityEqualOne();
16:         end if
17:     end if
18:     if(isNotCompositePrimaryKey(arrTables.get(i)))
19:         if(isNotPrimaryKey(cp.getColName()) && isFKeys(cp.getColName()))
20:             objectProperty=createObjectProperty(parentClass, childClass)
21:         end if
22:     end if
23:     if(isPrimaryKeyComposite(arrTables.get(i)) && getNumberOfPKeys() > getNumberOfFKs())
24:         p=listAllPrimaryKeysOfSpecifiedTable(arrTables.get(i))
25:         for each k<p.size() start
26:             if(isForeignKey(arrTables.get(i)), cp.getColName())
27:                 objectProperty=createObjectProperty(parentClass, childClass)
28:             else
29:                 createOWLDataPropertyWithMinCadrinalityEqualOne()
30:             end for
31:         end if
32: end

```

Le processus automatisé de notre algorithme reçoit en entrée le fichier SQL DLL (Din 2014) qui contient la définition de la RDB et génère le fichier OWL en sortie. Plus précisément, l'algorithme 1 obtient tous les modèles RDB décrits dans le tableau 2 puis il fait correspondre chaque élément RDB avec son élément correspondant dans OWL. Il est important de mentionner que notre algorithme est entièrement automatique. L'implémentation de cet algorithme est téléchargée dans notre dépôt GitHub.

2.3. Algorithme pour construire le ABOX

La méthode que nous traitons pour générer l'ABox est illustrée à la Fig. 1, notre méthode ne consiste pas seulement à générer l'ontologie à partir de la RDB mais également à valider le processus de cartographie lui-même (Plus de détails sur la validation au chapitre 4). Par conséquent, notre méthode est composée principalement d'un moteur d'extraction de métadonnées de base de données (DMEE), d'un moteur de génération de mappage (MGE) et d'un moteur R2RML.

Tout d'abord, le système reçoit en entrée un fichier SQL qui contient des instructions représentées par le langage DDL (Data Definition Language). Ensuite, le DMEE analyse ce fichier et en extrait automatiquement les métadonnées. Les métadonnées extraites comprennent les tables, les colonnes, les clés primaires (PKs) et les clés étrangères (FKs). Ensuite, MGE exploite les métadonnées extraites et construit un fichier de mapping (fichier R2RML). Enfin, le moteur R2RML prend en entrée le modèle de base de données (Schema + Instances) et le document de mappage généré qui contient un ensemble de règles représentant le schéma de base de données, puis fournit une sortie représentant le jeu de données RDF (triples). Plusieurs outils permettant de générer une ontologie à partir du fichier de correspondance R2RML et du modèle de base de données sont proposés dans la littérature, tels que morph-RDB [24] et Nknos [25], etc. Le moteur R2RML utilisé dans notre approche est r2rml-kit-master..

2.3.1. Extraction de Meta-Données (DMEE)

Notre processus de génération de métadonnées à partir d'une base de données relationnelle se compose de quatre étapes :

- Étape 1 : Identification des informations de la table. Nous introduisons la méthode `getTableInfo` qui consiste à extraire le nom des tables résidant dans la base de données relationnelle.
- Étape 2 : Identification des informations sur les colonnes. La méthode `getColumnInfo` est programmée pour extraire les informations de la colonne telles que le nom de la colonne, le type, et vérifier si la colonne n'est pas nulle pour chaque colonne.
- Étape 3 : Identification des clés primaires. La méthode `getPrimaryKeys` retourne pour chaque table, une liste de ses clés primaires.
- Étape 4 : Identification des clés étrangères. La méthode `getForeignKey` retourne pour chaque table, la liste de ses clés étrangères.

1.1.1. Générateur de BDR (MGE)

Dans cette phase, les cartes de triplets R2RML qui correspondent aux métadonnées identifiées dans la phase précédente sont produites. De manière concise, les principales étapes de mise en œuvre du MGE sont décrites dans l'algorithme suivant.

Algorithm 2: The sketch algorithm of generating the A-Box

```
Input : metadata;  
output: R2RML Mapping file;  
var:  
    tmap: TriplesMap;  
    collectionMap: Collection<TriplesMap>;  
    metadata: DBUtils;  
    table_names: ArrayList<String>;  
    colInfo: Map<String, List<TableComponent>>  
Begin  
1: table_names←metadata.getTablesInfo();  
2: for each i<table_names.size();  
3:     logicalTable←mf.createTableOrView()  
4:     colInfo←data.getColumnInfo(tbl_names.get(i))  
5:     for each key in colInfo.keySet() do  
6:         for each k < colInfo.get(key).size()  
7:             if(metadata.isPrimarykey()) then  
8:                 templet←mf.createTemplate()  
9:                 subjecMap←mf.createSubjectMap(templet)  
10:            else  
11:                pm←mf.createPredicateMap("ex:"+colInfo.getColName());  
12:                objectMap←mf.createObjectMap(colInfo.getColumnname());  
13:                prdicatObjectMap←mf.createPredicObjectMap(pm,oMap);  
14:                triplesMap←mf.createTriplesMap(ltable,sMap,pOMap);  
15:                if(metadata.isForeignKey())  
16:                    tmap←getRefObjetMap(getPtb, getPtC,getCht(),getChC());  
17:                triplesMap.setResource(createResource(url));  
18:                collectionMap.add(triplesMap);  
19:            R2RMLFileMapping←write(collectionMap)  
20: End
```

- LogicalTable : est la table SQL à partir de laquelle les triples RDF seront générés. La table logique peut être soit une table SQL, soit une vue SQL, soit une vue R2RML.

- Template : un template consiste en une série de segments de chaîne et de noms de colonne. Un modèle est utile pour créer des URIs basés sur les valeurs d'une ou plusieurs colonnes.

- SubjectMap : Une carte de sujets qui génère le sujet de tous les triples RDF qui seront générés à partir d'une ligne de table logique. Les sujets sont souvent des IRI générés à partir de la ou des colonnes clés primaires de la table.
- PredicateMap : Une carte de prédicat fait correspondre une colonne de la table à un prédicat dans le graphe RDF. Il peut y avoir plusieurs cartes de prédicats afin de spécifier plus de prédicats.
- ObjectMap : Une carte d'objets fait correspondre les valeurs de la base de données aux objets du graphe RDF. Un mappage d'objet de référencement est utile pour mapper les relations de clé étrangère.
- PredicateObjectMap : Une carte d'objets prédicats spécifie les prédicats et les objets des triplets RDF qui seront générés. Une carte d'objets de prédicats comporte au moins une carte de prédicats et une carte d'objets.
- TriplesMap : La carte de triples est la partie principale d'un mappage R2RML. Une carte de triples spécifie les triples RDF correspondant à une table logique.
- DBUtils : Est la classe qui contient le DMME.

1. Résultats et Discussion

Pour évaluer l'efficacité et la solidité du processus proposé, nous sommes partis de 6 bases de données relationnelles du domaine du commerce électronique. Ces bases de données couvrent plusieurs métadonnées utilisées dans le processus d'apprentissage d'ontologies à partir de bases de données relationnelles, comme les tables, les colonnes, les clés étrangères (FK) et les clés primaires (PK). Les informations détaillées de ces bases de données sont résumées dans le tableau 8. Comme preuve de concept, nos simulations expérimentales ont été menées sur un ordinateur personnel sous windows 10, avec un processeur Intel core i7 2.70 GHZ et 16 GB RAM.

RDB	Tables	Columns	PKs	FKs	Instances
North	30	150	30	25	5029
Iscommerce	5	20	5	6	200
Ecommerce	25	100	25	17	5000
EcommerceDB	3	20	4	2	1000
Sakila	16	90	18	22	47237
Northwind	13	89	16	13	2110

Table 3: A list of metadata extracted from RDB

1.1. La phase Discovery

Dans la phase de découverte, nous devons répondre à la question suivante : Avons-nous suffisamment d'informations de base pour commencer à construire une ontologie ? Le tableau 9 montre les questions les plus pertinentes que nous avons couvertes. Il peut être possible de se référer

à un expert dans le domaine étudié pour résoudre certains problèmes concernant les données recueillies, comme les problèmes de conceptualisation de la base de données (Spanos et al. 2012).

What is the domain that the ontology will cover?	E-Commerce domain
For what we are going to use the ontology (Application)?	Describing businesses, offering, prices, features, payments options, opening hours, and so on.
For what types of questions the information in the ontology should provide answers.	Table 5 depicts some competency questions for validating the resulting ontology.
What are the ontology intended uses and who are the end-users (Stakeholders)?	Customers, Employee, web Master, Accounting, etc.
What are the characteristics of the selected RDBs	Table 3 presents all the necessary metadata that we need to figure out the characteristics of the selected RDBs.
Is It necessary to interview the domain expert?	No

Table 4: The list of questions the ontologist must answer before start building ontology.

Contrairement à de nombreux processus traditionnels par étapes, dans lesquels le processus de construction d'une ontologie à partir de bases de données relationnelles commence sans vérifier si certains critères spécifiques sont remplis. Par conséquent, le cycle de vie proposé vise à tenir compte d'une plus grande ambiguïté. Comme le montre le tableau 4, il est recommandé de passer certains points de contrôle pour savoir si nous sommes prêts à passer à la phase suivante du cycle de vie de LOFRDB. La création d'un plan parfait pour l'apprentissage de l'ontologie à partir de la BDR nécessite une compréhension claire du domaine, du problème à résoudre et de l'étendue des sources de données à utiliser. Les réponses à ces questions clarifient la définition du problème et nous aident à sélectionner la base de données appropriée qui peut être utilisée dans les phases ultérieures. Le tableau 10 présente une liste de questions de compétence (CQ) qui représentent des questions informelles auxquelles l'ontologie doit pouvoir répondre (Pan et al. 2017a). Nous les considérons comme des phrases en langage naturel qui expriment des modèles pour les types de questions auxquelles les gens veulent pouvoir répondre avec l'ontologie.

Comme nous le savons, les auteurs d'ontologies sont généralement des experts du domaine mais ne maîtrisent pas nécessairement les technologies ontologiques, en particulier leurs fondements logiques (Pan et al. 2017a). Par conséquent, d'une part, il est difficile pour les auteurs humains d'exprimer leurs exigences pour l'axiomatisation d'une ontologie et, d'autre part, il est également difficile de savoir si les exigences sont remplies à la suite de leurs actions de création d'ontologie.

Pour résoudre ce problème, nous introduisons la méthodologie de la question de compétence afin d'aider les auteurs de l'ontologie à vérifier si l'ontologie résultante contient toutes les informations nécessaires. En fait, il est important de lister ces questions dans la phase de découverte afin de permettre à l'ontologiste de les prendre en considération au cours du processus de développement

Query 1: Find movie for a given set of generic features such as name and duration, etc.
Query 2: Retrieve basic information about a specific movie for display purposes.
Query 3: Find movie having a label that contains specific words.
Query 4: Get information about a reviewer
Query 5: Find movies having a label that contains specific words.
Query6: Find Text description of a given movie's title.
Query 7: Find movies that are similar to a given movies.

Table 5: A list of Competency Questions.

En outre, dans la phase de découverte, nous pouvons jeter un premier coup d'œil à la liste de données que nous avons choisie afin de déterminer si elle contient un grand nombre de métadonnées nécessaires. Le tableau 3 montre clairement que les bases de données relationnelles EcommerceDB et Iscommerce ne contiennent pas suffisamment de sémantique pour commencer à construire une ontologie. Par exemple, la base de données EcommerceDB contient 3 tables, 20 colonnes, 4 PKs, 2 FKs, et 100 instances. Sur la base de ces mesures, nous pouvons décider que la base de données EcommerceDB est pauvre sur le plan sémantique. Dans le même contexte, la base de données Iscommerce fournit également une sémantique pauvre. Par conséquent, dans la phase de découverte, nous pouvons supprimer les bases de données EcommerceDB et Iscommerce. Nous éliminons ces deux bases de données sur la base de la règle suivante : RDB pauvre sémantiquement implique ontologie pauvre sémantiquement (Konstantinou and Spanos 2015).

1.2. BDR Exploration

Maintenant, pour choisir le RDB le plus pertinent parmi ceux qui restent, nous devons calculer la mesure NS à partir des modèles décrits dans le tableau 11.

Rules	North	Ecommerce	Sakila	Northwind
Tables Without FKs	√	√	√	√
Tables With one FK	√	√	√	√
Tables with more than 2 FKs	√	√	√	√
Tables that contain exactly 2 FKs with presence of independent attribute	√	√	√	√

Many-to-many relationship: a table that contains exactly two FKs	√	√	√	√
FK+ NULL + Not UNIQUE	√	√	√	√
FK+ NOT NULL+ NOT UNIQUE	√	√	√	√
FK+ NOT NULL+ UNIQUE			√	√
FK+ NOT NULL+UNIQUE+ NOT PK			√	√
Attribute (neither PK nor FK	√	√	√	√
Attribute + NOT NULL	√	√	√	√
NOT FK + UNIQUE	√	√	√	√
PK	√	√	√	√
Attribute with constraint with an integer greater than 0			√	√
CHECK with enumeration			√	√
CHECK constraint as DataTypeRestriction			√	
The range of the foreign Key	√	√	√	√
Default Value			√	√
Two tables share the same primary key			√	
FK is a reference to the same table			√	
FK is a reference to the same table, but now it is accompanied by a trigger ON DELETE CASCADE			√	

Table 6: The set of patterns.

Comme indiqué précédemment, la métrique NS représente le nombre de caractéristiques sémantiques présentes dans la base de données relationnelle. Le tableau 6 montre que la base de données Sakila couvre toutes les sémantiques possibles qui peuvent être utilisées pour construire une ontologie riche à partir de RDB. Dans ce contexte, nous comparons également le nombre total de sémantiques par RDB comme indiqué dans le tableau 12. La première observation intéressante est que la base de données ayant un nombre total élevé de sémantiques ne signifie pas qu'elle couvre toutes les sémantiques possibles comme le montre le tableau 7. Par exemple, le nombre total de sémantiques pour la base de données Nord est de 180, mais le nombre de sémantiques est de 10 (moins de 17). Par conséquent, si nous décidons de construire une ontologie à partir des bases de données Nord et e-commerce, l'ontologie résultante ne traitera pas les sémantiques suivantes : Héritage, transitif, symétrique, restriction de valeur, restriction de plage de données, propriété fonctionnelle et fonctionnelle inverse. Cela permet de prédire que l'ontologie résultante basée sur les bases de données du Nord et du commerce électronique sera très pauvre sur le plan sémantique.

	North	E-commerce	Sakila	Northwind
NTDFK	5	12	5	4

NTFK	10	6	10	3
NTMTWOFK	8	7	6	3
NTEXTWOFK	7	4	7	3
NAFKNNU	15	10	7	5
NAFKNNNU	10	7	7	4
NAFKNNU	0	0	6	0
DefVal	0	0	3	0
NANNUNPK	0	0	2	0
NA	50	23	24	42
NAN	30	25	8	15
NANFKU	15	10	10	4
NPK	30	25	18	16
NACHECK	0	0	5	0
NADef	0	0	3	3
NTSAMEPK	0	0	2	0
NUnaryRel	0	0	2	0
NTrRel	0	0	1	0
Number of Semantics (NS)	10	10	17	13
Total Number of semantics	180	129	126	102

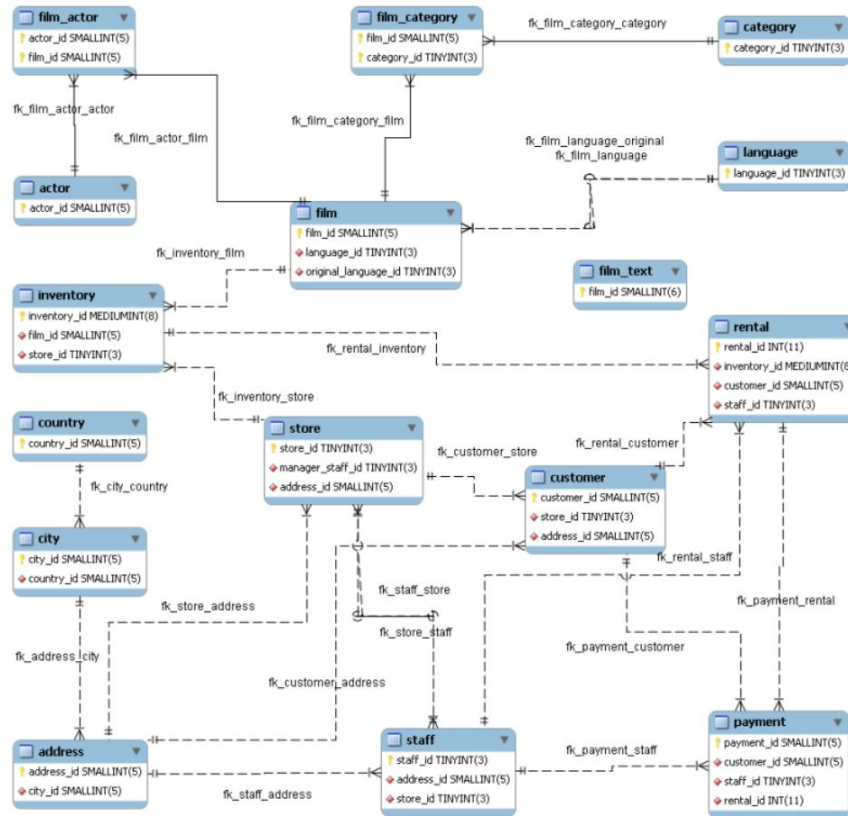
Table 7: The NS and the total number of semantics for each database.

Pour la base de données Northwind et Sakila, le NS et le nombre total de sémantiques sont (13,102) et (17,126) est 13 et son nombre total de sémantiques est 102. Nous pouvons remarquer que le nombre d'instances de chaque base de données est respectivement de 2120 et 47237 pour Northwind et Sakila. En conséquence, la base de données relationnelle la plus appropriée pour construire l'ontologie est Sakila, car elle couvre la sémantique la plus importante et le grand nombre d'instances.

1.3. De BDR à Ontologie: Sakila Database

La figure 13 montre le modèle conceptuel de la base de données Sakila. Comme nous l'avons déjà mentionné, l'entrée de nos algorithmes est le fichier SQL DDL qui est généré à partir de la couche physique. D'autre part, la figure 14 représente l'ontologie générée (T-Box) à partir de la base de

données Sakila. Par conséquent, au cours du développement, nous avons pris certaines décisions concernant la modélisation de l'ontologie, par exemple la table film_text n'est pas représentée comme une classe dans l'ontologie, nous l'avons transformée en attribut schema:description (owl:DataProperty) qui a une table Film comme domaine. Nous avons fait de même pour la table de langue. L'ontologie sakila résultante contient 12 classes au lieu des 18 tables qui sont représentées dans la base de données originale. En outre, Sakila a 14 propriétés d'objet et 25 propriétés de données. Par conséquent, nous pouvons en déduire que la modélisation ontologique peut également générer un modèle simple par rapport à ce que nous avons dans la base de données



relationnelle.

Figure 6: Sakila Database EER Diagram

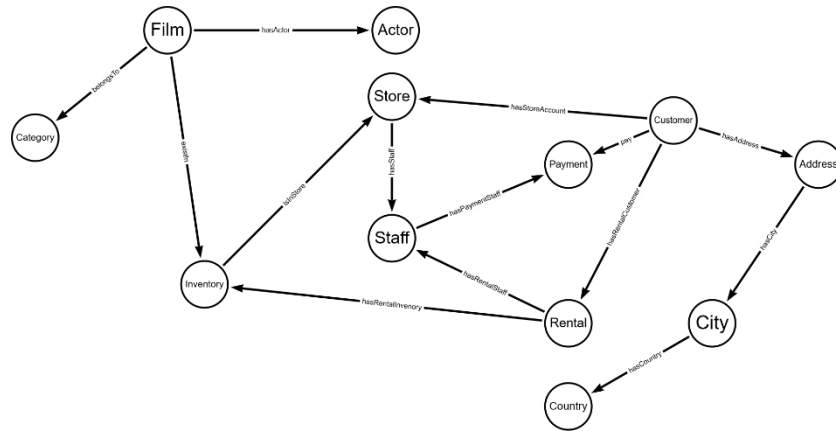


Figure 7: Sakila Ontology Constructed from sakila database

Conclusion

En résumé, dans ce chapitre, nous avons essayé de rassembler les approches les plus importantes et les plus contributives dans le domaine du mapping de la base de données relationnelle vers l'ontologie. Nous avons essayé de fournir au lecteur un aperçu concis de ces approches en identifiant les principaux inconvénients auxquels les chercheurs dans ce domaine sont confrontés et en suggérant des solutions. En outre, les plus grandes contributions de ce chapitre sont les suivantes :

- Nous proposons un nouveau cycle de vie pour l'apprentissage d'ontologies à partir de bases de données relationnelles, basé sur les exigences du génie logiciel. Comme nous l'avons constaté dans la littérature, l'apprentissage d'ontologies à partir de bases de données relationnelles n'identifie pas l'ordre dans lequel les activités et les étapes doivent être réalisées. Dans ce contexte, le rôle du cycle de vie de l'ontologie est d'identifier le moment où les activités doivent être réalisées, c'est-à-dire qu'il détermine l'ensemble des phases par lesquelles l'ontologie passe au cours de sa vie, décrit les activités à réaliser dans chaque étape et comment les étapes sont liées.
- Nous décrivons une nouvelle méthode pour construire une ontologie à partir d'une base de données relationnelle basée sur le cycle de vie prédéfini. Cette méthode comprend les phases suivantes
- L'exploration des RDB où nous pouvons vérifier si les bases de données relationnelles d'entrée contiennent l'espace complet des métadonnées et des caractéristiques sémantiques pour générer des ontologies. Dans ce contexte, nous présentons le paramètre nombre de sémantiques (NS) qui fournit les caractéristiques de chaque RDB d'entrée. En outre, l'exploration de la RDB nécessite également l'intervention humaine pour confirmer la validité des bases de données sélectionnées, car le NS n'est pas suffisant pour prendre cette décision, mais il reste utile.

- Construction de la TBox. La TBox fournit le vocabulaire d'un domaine d'application. Elle contient la définition des classes, des concepts, des propriétés et des axiomes. Afin de construire une TBox, un ensemble de patrons de transformation est fourni. Le processus de recherche de la correspondance entre la RDB et les modèles d'ontologie est automatique et utilise un ensemble d'algorithmes. En outre, nous proposons trois nouvelles règles de transformation qui permettent de convertir la contrainte de vérification, la contrainte par défaut et la contrainte d'amélioration de la relation d'héritage en correspondance avec les patrons ontologiques.

- Construction de l'ABox. L'ABox contient un ensemble d'assertions (instances). En fait, l'ABox gère le niveau d'extension de l'ontologie en s'appuyant sur le mappage pour accéder aux sources de données externes telles que les bases de données relationnelles dans notre cas. Le processus de génération de l'ABox est réalisé à l'aide du langage R2RML.

Comme nous l'avons vu dans le cycle de vie proposé, nous avons une phase d'évaluation où nous devons évaluer l'ontologie obtenue avant de la publier sur le Web. Dans ce contexte, les métriques que nous utilisons dans ce chapitre ne représentent qu'une ébauche et ne sont pas complètes pour prendre cette décision. Plus précisément, nous ne pouvons pas savoir si les ontologies que nous construisons ou utilisons sont bonnes tant que nous ne savons pas ce que signifie "bon" et comment nous pouvons le mesurer. Pour cela, dans le prochain chapitre, nous discuterons plus en profondeur des principales dimensions de la qualité que nous devons considérer chaque fois que vous jugez de la qualité d'une ontologie, ainsi que des principales métriques et méthodes que nous pouvons utiliser pour mesurer ces dimensions.

Chapiter2. L'Evaluation de L'ontologie:

Vérification Et Validation

Ce chapitre est divisé en deux axes principaux. Le premier axe est consacré à vérifier l'exactitude de l'ontologie construite où l'on vérifie si l'ontologie a été construite correctement. Dans le deuxième axe, nous présentons une analyse des métriques avancées pour mesurer la complexité de conception des ontologies du patrimoine culturel (CH) existantes afin d'aider les développeurs à décider si l'ontologie est trop complexe et nécessite une simplification ou une reconstruction et les développeurs réalisent clairement l'impact de la taille et de l'échelle de l'ontologie.

1. Les approches de l'évaluation

Diverses approches ont été utilisées pour l'évaluation des ontologies. Certaines d'entre elles sont l'évaluation de l'étalon-or, l'évaluation basée sur les données, l'évaluation basée sur les applications ou les tâches, l'évaluation basée sur l'utilisateur et l'évaluation basée sur les métriques.

1.1. Gold Standard

L'approche de l'étalon-or tente de comparer l'ontologie apprise avec une ontologie de référence précédemment créée, appelée étalon-or (Lourdusamy and John 2018b). Cet étalon-or peut nous indiquer à tout moment à quel point l'ontologie est proche de son achèvement. Cependant, disposer d'un étalon-or est un défi, car il doit être créé dans des conditions et avec des objectifs similaires à ceux de l'ontologie apprise. C'est pourquoi nous utilisons généralement des normes d'or partielles ou des normes d'argent. Un étalon-or partiel contient un sous-ensemble des connaissances que le modèle doit contenir. Par exemple, dans Farber et al. (Färber et al. 2018) les auteurs ont créé un étalon-or partiel avec 41 classes et 22 relations pour 5 domaines (personnes, médias, organisations, géographie et biologie) afin de mesurer et de comparer la complétude de DBpedia, YAGO et d'autres modèles sémantiques disponibles publiquement. En outre, Maedche et Staab (Maedche and Staab 2002) considèrent les ontologies comme des systèmes à deux niveaux, composés d'une couche lexicale et d'une couche conceptuelle. Sur la base de ce modèle d'ontologie de base, cette approche mesure la similarité entre l'ontologie apprise et une ontologie du domaine du tourisme modélisée par des experts. Elle mesure la similarité en se basant sur la notion de lexique, les fonctions de référence et la cotation sémantique qui sont décrites en détail dans (Maedche and Staab 2002).

1.2. Data Driven

Les approches basées sur les données, également connues sous le nom d'approches basées sur le corpus, sont utilisées pour évaluer dans quelle mesure une ontologie couvre suffisamment un domaine donné. Le concept de ce type d'approche consiste à comparer l'ontologie apprise avec le contenu d'un corpus textuel qui couvre de manière significative un domaine donné. L'avantage est de comparer une ou plusieurs ontologies avec un corpus, plutôt que de comparer une ontologie avec une autre existante. Une approche de base consiste à effectuer une extraction automatique de termes sur le corpus et à compter simplement le nombre de concepts qui se chevauchent entre l'ontologie et le corpus. Une autre approche consiste à utiliser une représentation en espace vectoriel des concepts à la fois dans le corpus et dans l'ontologie évaluée afin de mesurer l'adéquation entre eux. En outre, (Brewster et al. 2004) évaluent l'ontologie apprise en appliquant d'abord l'analyse sémantique latente et les méthodes de regroupement pour identifier les mots-clés dans un corpus. Comme chaque mot-clé peut être représenté d'une manière lexicale différente, cette approche utilise WordNet pour étendre les requêtes. Enfin, l'ontologie peut être évaluée en mettant en correspondance l'ensemble des concepts identifiés dans le corpus avec l'ontologie apprise. Comme ce type d'approche d'évaluation peut être considéré comme similaire à bien des égards à l'approche basée sur l'étalon-or. En outre, le principal défi de ce type d'approche est également similaire à celui des approches basées sur l'étalon-or. Cependant, il est plus facile. Trouver un corpus qui couvre le même domaine que l'ontologie apprise est nettement plus facile que de trouver une ontologie spécifique au domaine bien représentée. Par exemple, Jones et Alani (Jones and Alani 2006) utilisent le moteur de recherche Google pour trouver un corpus basé sur une requête d'utilisateur. Après avoir étendu la requête de l'utilisateur en utilisant WordNet, les 100 premières pages des résultats de Google sont considérées comme le corpus pour l'évaluation.

1.3. Task-Based

Les approches basées sur les tâches tentent de mesurer dans quelle mesure une ontologie contribue à améliorer les résultats d'une certaine tâche. Ce type d'évaluation considère qu'une ontologie donnée est destinée à une tâche particulière, et est évaluée uniquement en fonction de ses performances dans cette tâche, indépendamment de toutes les caractéristiques structurelles. Par exemple, si l'on conçoit une ontologie pour améliorer les performances d'un moteur de recherche Web, on peut collecter plusieurs exemples de requêtes et comparer si les résultats de la recherche contiennent plus de documents pertinents si une certaine ontologie est utilisée (Brank et al. 2005). Les approches basées sur les tâches sont considérées comme les plus efficaces pour évaluer l'adaptabilité d'une ontologie, en appliquant l'ontologie à plusieurs tâches et en évaluant ses performances pour ces tâches. En outre, les approches basées sur les tâches sont principalement utilisées pour évaluer la compatibilité entre l'outil utilisé et l'ontologie, et pour calculer la vitesse d'exécution de la tâche prévue. Cette approche présente une limite : le résultat obtenu pour une tâche peut ne pas être utile pour une autre tâche car chaque tâche est différente. Les méthodes

utilisées dans l'évaluation basée sur les tâches sont destinées à un contexte spécifique et peuvent donc ne pas fournir de résultats précis dans un autre contexte. Ainsi, cette approche est adaptée à un petit nombre d'ontologies plutôt qu'à un grand nombre (Lourdusamy and John 2018b).

1.4. User Based

L'évaluation basée sur l'utilisateur évalue les ontologies en fonction de l'expérience de l'utilisateur. Bien que la contribution des utilisateurs puisse être bénéfique, en raison de leurs différences de connaissances et d'expérience, tous les utilisateurs ne peuvent pas être considérés au même niveau. Parfois, l'évaluation des utilisateurs peut être subjective

1.5. Metric Based

L'approche basée sur les métriques offre une perspective quantitative de la qualité des ontologies. Dans cette approche, différentes métriques sont utilisées pour évaluer la qualité et l'adéquation de l'ontologie dans un domaine spécifique (Brank et al. 2005). Différentes méthodes sont utilisées pour assimiler les informations de l'ontologie et sur l'ontologie ; divers outils utilisent différentes métriques pour l'évaluation.

2. Les categories de l'évaluation

Le processus d'évaluation est divisé en deux domaines principaux : la vérification et la validation.

2.1. La verification de l'ontologie

La vérification est la tâche qui consiste à évaluer si l'ontologie a été construite correctement. Elle vérifie l'encodage de la spécification. Des erreurs telles que des hiérarchies de classes circulaires, des axiomes redondants, des schémas de dénomination incohérents, etc. La vérification confirme que l'ontologie a été construite conformément à certains critères de qualité spécifiés. Plus profondément, la vérification s'intéresse à la construction correcte d'une ontologie en mesurant les métriques d'exactitude, de complétude, de concision et de cohérence, etc (Pak and Zhou 2009).

2.2. La Validation de l'ontologie

La validation est la tâche consistant à évaluer si l'ontologie correcte a été construite. La validation consiste à déterminer si le sens des définitions correspond à la conceptualisation que l'ontologie est censée spécifier. L'objectif est de montrer que le modèle du monde est conforme aux modèles formels. La validation consiste à construire l'ontologie correcte en vérifiant la qualité de la conception de l'ontologie. La conception de l'ontologie est communément appelée la complexité de l'ontologie (Pak and Zhou 2009). La validation mesure la complexité d'une ontologie donne un aperçu aux développeurs pour les aider à mieux comprendre, réutiliser, réduire les exigences de

maintenance et intégrer les ontologies, ainsi que pour aider les utilisateurs à sélectionner l'ontologie qui répond le mieux à leurs besoins.

En bref, pour mesurer la validation et la vérification d'une ontologie spécifique, nous avons adopté l'approche basée sur les métriques combinée à l'évaluation basée sur l'utilisateur. Il semble que ces deux approches soient les plus efficaces dans l'évaluation. La raison de cette combinaison est que l'approche métrique ne nous fournit pas de seuils standardisés pour prendre des décisions. Dans ce cas, l'expérience de l'utilisateur est obligatoire. Parfois, l'évaluation des utilisateurs peut être subjective.

3. La vérification de l'ontologie générée à partir de la base de données relationnelle

Dans cette section, nous aborderons les questions de la vérification de l'ontologie. Dans ce contexte, nous avons divisé le processus d'évaluation en deux catégories : l'évaluation de la T-Box et l'évaluation de la A-Box.

3.1. L' évaluation de TBox

Bien que nous ne puissions pas vraiment savoir si la conception de la T-Box modélise correctement la connaissance du domaine, des métriques telles que la richesse, la largeur, la profondeur et l'héritage indiquent la qualité de la T-Box créée. Par conséquent, les mesures les plus importantes de cette catégorie sont décrites ci-dessous.

3.1.1. Attribute Richness (AR)

AR représente le nombre moyen d'attributs (emplacements) par classe. Généralement, nous supposons que plus les attributs sont générés à partir de RDB, plus les connaissances sont transmises à l'ontologie (Lourdusamy and John 2018a).

Definition. La richesse attributive est définie comme le nombre moyen d'attributs par classe. Elle est calculée comme le nombre d'attributs pour toutes les classes (ATT) divisé par le nombre de classes (C).

$$AR = \frac{|ATT|}{|C|}$$

3.1.2. Inheritance Richness (IR)

Cette métrique représente la distribution des informations à travers les différents niveaux de la T-Box et sert d'indicateur de la façon dont les connaissances sont regroupées en différentes catégories et sous-catégories dans la TBox. Une TBox avec un IR faible indique que la T-Box couvre un

domaine spécifique de manière détaillée, tandis qu'une T-Box avec un IR élevé représente une connaissance générale (Lourdusamy and John 2018a).

Definition. IR est défini comme le nombre moyen de sous-classes par classe, où H est la somme du nombre de relations d'héritage et C est le nombre total de classes

$$IR = \frac{|H|}{|C|}$$

3.1.3. Relationship Richness (RR)

Cette métrique reflète la diversité des types de relations dans la TBox, par exemple. Une TBox qui ne contient que des relations d'héritage transmet généralement moins d'informations qu'une TBox qui contient un ensemble diversifié de relations telles que des relations transitives, symétriques et réflexives (Tartir et al. 2005).

Definition. La RR d'une boîte T est définie comme le rapport du nombre de relations de non-héritage (P), divisé par la somme des relations d'héritage (H) et des relations de non-héritage (P).

$$RR = \frac{|P|}{|H| + |P|}$$

3.2. La validation A-Box

Les métriques d'évaluation de la boîte à outils peuvent être utilisées pour vérifier comment les données sont placées dans l'ontologie. Plus précisément, l'évaluation de la boîte à outils fait référence aux métriques des instances. À cet égard, nous avons utilisé deux métriques prédéfinies : La richesse des classes et la population moyenne.

3.2.1. Class Richness (CR)

CR est lié à la façon dont les instances sont réparties entre les classes. Le nombre de classes qui ont des instances dans la KB est comparé au nombre total de classes, ce qui donne une idée générale de la façon dont la KB utilise les connaissances modélisées par la T-Box. Une boîte A avec un faible CR indique que la boîte A n'a pas de données qui illustrent toutes les connaissances de classe qui existent dans la boîte T. D'autre part, une boîte A avec un CR élevé prouve que les données dans la boîte A couvrent la plupart des connaissances (Lourdusamy and John 2018a).

Definition. CR est défini comme le rapport entre le nombre total de classes qui ont des instances c' divisé par le nombre total de classes(C).

$$CR = \frac{|c'|}{|C|}$$

3.2.2. Average Population (AP)

Cette mesure est une indication du nombre d'instances par rapport au nombre de classes. Elle peut être utile si le développeur d'ontologie n'est pas sûr que suffisamment d'instances ont été extraites par rapport au nombre de classes (Lourdusamy and John 2018a).

Definition. *AP* est défini comme le nombre d'instances de A-Box (I) divisé par le nombre de classes définies dans le schéma de l'ontologie (C).

$$AP = \frac{|I|}{|C|}$$

4. Métriques de complexité avancées de l'ontologie

4.1. Métriques de complexité avancées de l'ontologie

Les mesures de complexité avancées utilisées dans ce travail comprennent : la taille du vocabulaire (SOV), l'impureté des arbres, le nombre moyen de chemins par concept, la longueur moyenne des chemins de l'ontologie et le couplage (Zhang et al. 2010).

4.1.1. SOV

SOV mesure la taille du vocabulaire en utilisant des métriques primitives telles que le nombre de classes. Étant donné un graphe $G = \langle N, P, E \rangle$ d'une ontologie, SOV est défini comme la somme des classes nommées et des individus nommés (N_n) et des propriétés définies par l'utilisateur (P_n):

$$sov = |N_n| + |P_n|$$

Un SOV plus élevé implique que l'ontologie est de grande taille et nécessiterait beaucoup de temps et d'efforts pour la construire et la maintenir.

4.1.2. Tree impurity (TIP)

Cette métrique est utilisée pour mesurer dans quelle mesure une hiérarchie d'héritage d'ontologie s'écarte d'un arbre $G' = \langle N', P', E' \rangle$ (Zhang et al. 2010). Il est défini comme:

$$TIP = |E'| - |N'| + 1$$

Où E' est le nombre d'arêtes `rdfs:subClassOf` et N' est le nombre de nœuds (y compris les nœuds nommés et anonymes) dans la hiérarchie d'héritage d'une ontologie. Plus le TIP est élevé, plus la hiérarchie d'héritage d'une ontologie s'écarte d'une structure arborescente pure, et plus la complexité d'une ontologie est grande. A $TIP = 0$ signifie que la hiérarchie d'héritage est un arbre.

4.1.3. Average number of paths per concept

The average number of paths per concept (ρ) - indique le degré de connectivité moyen d'un concept au concept racine dans la hiérarchie d'héritage de l'ontologie (Lourdusamy and John 2018b). Il est défini comme le rapport entre le nombre total de chemins et le nombre total de classes (m):

$$\rho = \frac{\sum_{i=1}^m p_i}{m}$$

Pour toute ontologie, ρ doit être supérieur ou égal à 1 (chaque concept doit avoir un parent sauf le concept général). Si $\rho = 1$, alors l'ontologie est un arbre (chaque concept a un seul parent, et donc un seul chemin vers le concept le plus général). Une ontologie avec un ρ plus élevé indique que les changements dans une classe auraient un impact important sur ses sous-classes (chaque concept a plusieurs parents, et donc plusieurs chemins vers le concept le plus général).

4.1.4. The average path length

The average path length ($\bar{\Lambda}$) indique le nombre moyen de concepts dans un chemin de l'ontologie (Zhe et al. 2006). Elle est définie comme suit:

$$\bar{\Lambda} = \frac{\sum_{i=1}^m \sum_{k=1}^{p_i} p l_{i,k}}{\sum_{i=1}^m p_i}$$

Cette métrique est obtenue à partir du rapport de la somme des longueurs des chemins ($p l_{i,k}$) de chacun des m concepts de l'ontologie sur la somme du nombre de chemins (p_i) des concepts. Une ontologie avec un $\bar{\Lambda}$ plus grand indique qu'il y a trop de relations d'héritage dans l'ontologie ; en conséquence, la gestion et la manipulation des concepts dans une telle ontologie pourraient être une tâche complexe.

4.1.5. Coupling

Le couplage reflète le nombre de classes externes provenant d'ontologies importées qui sont référencées dans l'ontologie interne (locale). Tout comme la mesure du couplage des modules logiciels, le couplage des ontologies mesure la parenté de l'ontologie locale avec d'autres ontologies ou vocabulaires existants utilisés pour construire cette ontologie (Ouyang et al. 2011). Il est défini comme suit :

$$Coup(O_i) = 1 - \frac{Ref(O_i)}{NEC(O_i)}$$

Où $Ref(O_i)$ le nombre de classes externes est référencé et $NEC(O_i)$ représente le nombre de classes externes. Plus le couplage est fort dans les ontologies, plus les systèmes qui utilisent ces ontologies sont difficiles à comprendre, à maintenir et plus ils sont complexes.

5. Expériences d'évaluation d'ontologie

5.1. La verification de l'ontologie

Pour la vérification de la construction de l'ontologie, nous avons généralement comparé notre ontologie résultante à une norme d'or qui est conçue de manière appropriée pour le domaine du discours (Hlomani and Stacey 2014). Il peut s'agir en fait d'une ontologie considérée comme bien construite pour servir de référence. Comme nous l'avons déjà mentionné, le domaine du discours que nous traitons est le commerce électronique (ORDYSSISKI 2011). L'ontologie de référence qui représente le domaine du commerce électronique est l'ontologie GoodRelations (Hepp 2008b). Il s'agit d'un vocabulaire normalisé pour les données relatives aux produits, aux prix et aux entreprises qui peuvent être (i) intégrées dans des pages Web existantes et dynamiques et (ii) traitées par d'autres ordinateurs. En général, GoodRelations est utilisé pour faciliter la création de descriptions formelles de l'offre de produits pour le commerce électronique. Le tableau 13 montre les métriques de base de l'ontologie GoodRelations par rapport à l'ontologie résultante.

Ontology	TCC	Axioms	LAC	TOP	TDP	TINDV
GoodRelations	38	1141	450	53	49	46
Resulting ontology	23	28816	28816	60	72	47803

Table 8: The basic metrics of the GoodRelations versus the resulting ontology.

Les métriques de base de l'ontologie fournissent le nombre de classes, d'objets, d'axiomes, de propriétés et d'instances utilisés dans l'ontologie. Si l'on considère le résultat présenté dans la Figure 8, on voit clairement que notre ontologie résultante couvre plus de connaissances de base que l'ontologie de référence en ce qui concerne le nombre total de classes, le nombre total de propriétés de type de données (TDP) et d'objets (TOP), le nombre d'axiomes logiques (LAC), le nombre d'axiomes et le nombre d'instances (TINDV). Par exemple, le TINDV est de 47803 et 46 pour l'ontologie résultante et l'ontologie de référence respectivement. Cependant, nous ne pouvons pas discuter de la qualité de l'ontologie résultante sur la base de ces métriques, car celles-ci ne représentent que l'effet discriminant de la couverture des connaissances (Hlomani and Stacey 2014) comme le montre la figure 13. A cet égard, les deux sous-sections suivantes expliquent bien les métriques que nous avons utilisées pour mesurer la qualité de notre ontologie.

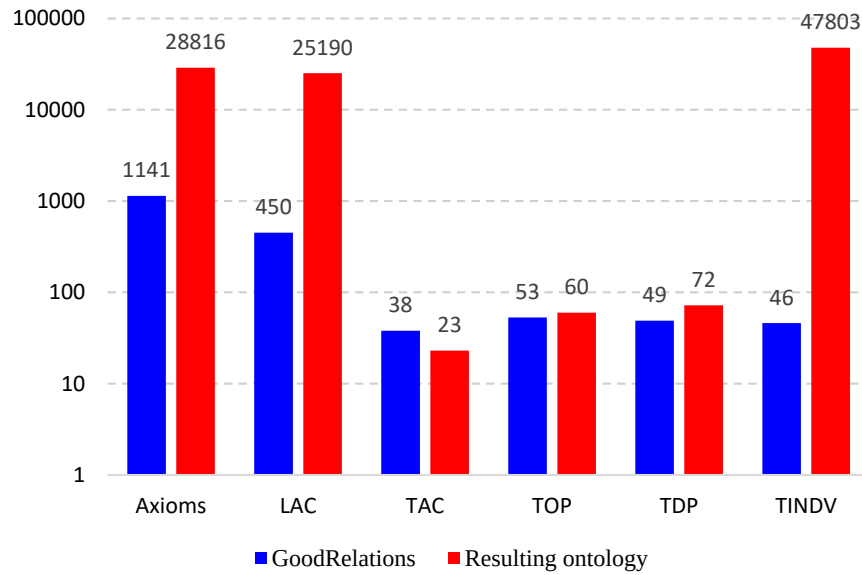


Figure 8: Discriminative effect of the knowledge coverage.

5.1.1. L'évaluation TBox

Les valeurs de RI proches de zéro indiquent une ontologie plate ou horizontale représentant peut-être des connaissances plus générales, tandis que les grandes valeurs représentent des ontologies verticales décrivant des connaissances détaillées d'un domaine. Comme le montre le tableau 9, l'IR de notre ontologie est de 2,357 alors que celui de GoodRelations est de 0,5. Cela indique que notre ontologie décrit mieux le domaine du commerce électronique que l'ontologie de référence. Cependant, la richesse des relations pour l'ontologie de référence est plus grande que l'ontologie résultante, ce qui indique que l'ontologie de référence contient de nombreuses relations autres que les relations classe-sous-classe, alors que notre ontologie est plus riche qu'une taxonomie avec seulement des relations classe-sous-classe. D'autre part, la richesse en attributs de notre ontologie résultante est significativement plus grande que l'AR de l'ontologie de référence, ce qui indique que notre ontologie définit plus de connaissances que l'ontologie de référence.

	A-Box Evaluation		T-Box Evaluation		
Ontology	AP	CR	AR	IR	RR
GoodRelations	1,21	0,236	1,89	0,5	0,9082
Resulting ontology	2078,39	0,7142	5,142	2,357	0,3125

Table 9: The list of metrics for evaluating the resulting ontology.

Selon le résultat décrit dans le tableau 14, nous avons présenté la sortie TBox de chaque approche étudiée en utilisant un élément OWL spécifique. En outre, la dernière colonne montre le résultat de notre mise en correspondance. Il est évident, d'après ce tableau, que notre ontologie contient un nombre élevé de sémantiques par rapport aux autres approches.

1.1.1. L'évaluation ABox

Le premier groupe de mesures que nous avons considéré pour cette validation est lié à la distribution des connaissances dans l'ontologie. Comme nous pouvons le voir dans le tableau 14, la population moyenne (AP) de l'ontologie résultante est meilleure que celle de l'ontologie de référence. Par rapport à l'ontologie de référence, la valeur de la PA de notre ontologie, qui est de 2078.39, implique que notre ontologie offre un nombre suffisant d'instances pour décrire le domaine du commerce électronique. Selon les auteurs de (Sicilia et al. 2012), cette métrique est proposée pour être utilisée en conjonction avec la métrique de richesse des classes (CR). À cet égard, nous avons calculé la métrique CR. La valeur de cette métrique confirme que les classes de notre ontologie sont peuplées d'un nombre élevé d'instances par rapport à l'ontologie GoodRelations, ce qui reflète la diversité des connaissances intégrées dans notre boîte à outils.

1.2. La validation de l'ontology

Une série d'expériences appropriées ont été organisées afin d'étudier la complexité des ontologies bien connues du patrimoine culturel sélectionnées. Les informations détaillées de ces ensembles de données sont résumées dans le tableau 1. Comme preuve de concept, les métriques de complexité avancées sont calculées à l'aide de l'API Java OWL (Horridge & Bechhofer, 2011). Enfin, il est important de noter que toutes les simulations expérimentales ont été réalisées sur un ordinateur personnel sous Windows 10, avec processeur Intel Core i7 2,70 GHz et 16 Go de RAM.

1.2.1. Datasets

La base de données est composée de 20 ontologies du domaine CH. Chaque ontologie du jeu de données se voit attribuer un index O_i , $1 \leq i \leq 20$ pour faciliter sa référence dans la discussion. Le tableau 15 montre la liste des ontologies dans l'ensemble de données avec leurs noms et leurs liens Web. Les fichiers XML sont des documents Web qui incluent les fichiers RDF/OWL des ontologies correspondantes.

Index	Ontology	Category	Web Link
O_1	FRBR	Actor	https://vocab.org/frbr/core
O_2	Hico	Event	http://hico.sourceforge.net/
O_3	Bio	Actor	https://vocab.org/bio/
O_4	Cito	Event	https://w3id.org/spar/cito/
O_5	Pro	Event	https://w3id.org/spar/pro/

O_6	bibo	Actor	http://bibliontology.com/
O_7	Fabio	Actor	https://lov.linkeddata.es/dataset/lov/vocabs/fabio
O_8	Cidoc CRM	Event+Actor+Place	http://www.cidoc-crm.org/
O_9	Cultural-event	Event	https://lov.linkeddata.es/dataset/lov/vocabs/cevent
O_{10}	Cultural-On	Event	https://lov.linkeddata.es/dataset/lov/vocabs/cis
O_{11}	Event	Event	https://lov.linkeddata.es/dataset/lov/vocabs/event
O_{12}	SEAS	Event	https://lov.linkeddata.es/dataset/lov/vocabs/seas
O_{13}	SEM	Event	https://lov.linkeddata.es/dataset/lov/vocabs/sem
O_{14}	Tp	Place	https://lov.linkeddata.es/dataset/lov/vocabs/tp
O_{15}	DOLCE-lite	Event	http://www.ontologydesignpatterns.org/ont/dlp/DOLCE-Lite.owl
O_{16}	GVP	Event+Place	https://lov.linkeddata.es/dataset/lov/vocabs/gvp
O_{17}	Ctlog	Event	https://lov.linkeddata.es/dataset/lov/vocabs/ctlog
O_{18}	Cdesc	Event+Place+Actor	https://lov.linkeddata.es/dataset/lov/vocabs/cdesc
O_{19}	Drama	Event+Actor+Place	https://lov.linkeddata.es/dataset/lov/vocabs/drama
O_{20}	ddesc	Event+Place+Actor	https://lov.linkeddata.es/dataset/lov/vocabs/ddesc

Table 10: the studied CH ontologies.

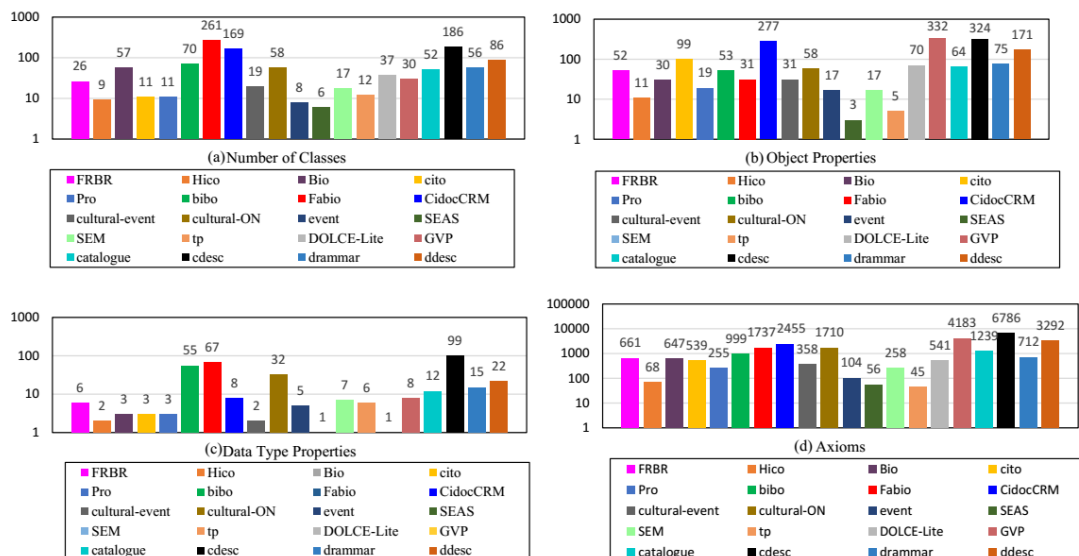


Figure 9: The primitive metrics of the studied CH ontologies.

1.2.2. Result and Discussions

L'objectif principal de ce travail est d'analyser les caractéristiques de complexité avancée des ontologies CH et leur impact sur l'évolution et la réutilisation des ontologies. Dans ce contexte, chacune des mesures de complexité avancée est calculée et discutée dans les sections suivantes.

1.2.2.1. Size Of The Vocabulary (SOV)

La figure 15 présente les résultats de la mesure du SOV pour toutes les ontologies de l'ensemble de données. Le SOV varie de 11 à 655, ce qui montre les différentes quantités de vocabulaire utilisées. La majorité des ontologies ont un SOV compris entre 50 et 700, suivies par celles dont le SOV est compris entre 20 et 40. Ces résultats indiquent qu'il serait avantageux pour les développeurs de Web sémantique dans le domaine CH d'envisager la réutilisation de ces ontologies (ontologie bibliographique (O_7 , SOV=374), ontologie d'événement (O_8 , SOV=454), ontologie de lieu (O_{16} and O_{20} , SOV=430 and 372) et ontologie générale (O_{18} , SOV=655)) plutôt que d'essayer de construire de novo de nouvelles ontologies connexes. La SOV de ces ontologies indique également qu'il faudrait plus de temps et d'efforts pour les reconstruire et les maintenir (Zhang et al. 2010).

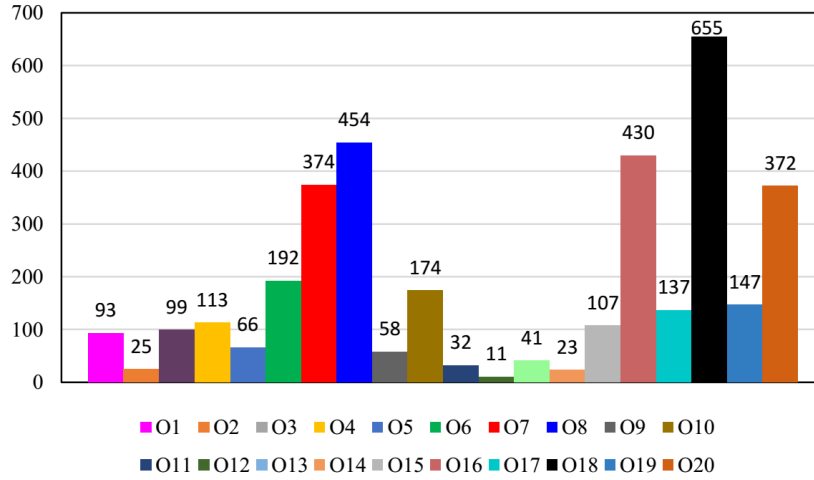


Figure 10: The Size of Vocabulary of the studied ontologies.

1.2.2.2. Average path length ($\bar{\Lambda}$), and Average Number of Paths per Concept (ρ)

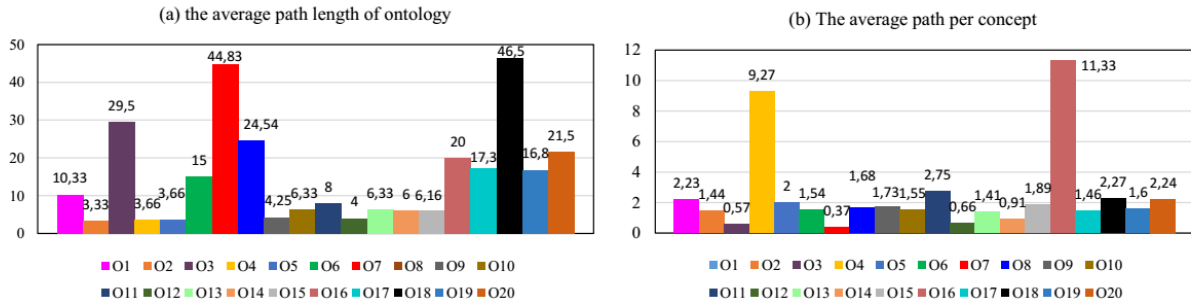


Figure 11: the average path length and number of Paths per concepts.

La figure 16 présente une analyse conjointe de la longueur moyenne des chemins de l'ontologie et du nombre moyen de chemins par concept. Ces deux métriques pour toutes les ontologies sont regroupées en 2 plages. Les deux plages pour $\bar{\Lambda}$ sont: $0 \leq \bar{\Lambda} < 10$ et $10 \leq \bar{\Lambda} \leq 50$. La figure 3(a) montre que la majorité des ontologies ont $\bar{\Lambda}$ dans la plage $0 \leq \bar{\Lambda} < 50$, tandis que d'autres sont dans la première plage. Cela indique que les ontologies ont un $\bar{\Lambda}$ dans la plage $0 \leq \bar{\Lambda} < 50$, andis que d'autres sont dans la première plage. Cela indique que les ontologies ont un $\bar{\Lambda}$ élevé. Par conséquent, un $\bar{\Lambda}$ plus élevé montre que la classe réside plus profondément dans la hiérarchie d'héritage et réutilise plus d'informations de ses ancêtres tels que o_{18} , o_7 , o_8 , etc. Un $\bar{\Lambda}$ élevé indique également que la classe est plus difficile à maintenir car elle est susceptible d'être affectée par des changements dans n'importe lequel de ses ancêtres. Dans le même contexte, les deux plages pour ρ sont: $\rho < 1$ and $\rho \geq 1$. À partir de l'analyse de la valeur de ρ (Figure 3 (b)), on peut confirmer que la plupart des ontologies dans les ensembles de données ont un chemin multiple de la classe racine à une classe donnée, ce qui indique que presque toutes les ontologies s'appuie sur le type de modèle réseau (graphe) plutôt que sur le type de modèle hiérarchique (Arbre). Plus précisément,

les ontologies avec un petit ρ ($\rho < 1$) indiquent que les changements dans une classe auraient un impact moindre sur ses sous-classes (Zhe et al. 2006).

1.2.2.3. Tree Impurity (TIP)

La figure 17 présente les résultats du calcul du TIP pour toutes les ontologies de l'ensemble de données. On remarque qu'un nombre important d'ontologies ont un TIP compris entre 100 et 2500, suivi par celles dont le TIP est inférieur à 40. Ce résultat empirique montre que la plupart des ontologies adoptent l'héritage multiple (TIP >> 0) et que leur hiérarchie d'héritage s'écarte fortement d'une structure arborescente pure ($o_7 = 1377$, $o_{13} = 2295$, $o_{20} = 2446$, etc.). Plus précisément, cela indique que ces ontologies ne peuvent pas être facilement maintenues à l'exception de o_2 , o_{11} , o_{14} qui ont respectivement TIP 42, 21, 20, 29.

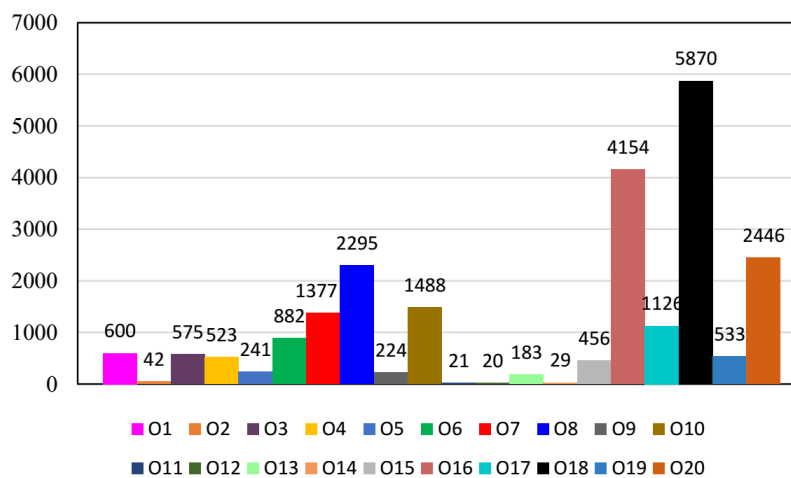


Figure 12: The Tree Impurity of the studied ontologies.

1.2.2.4. Coupling

En considérant le résultat présenté dans la Figure 18, on voit clairement que le couplage des ontologies suivantes : o_8 , o_9 , o_{11} , o_{15} , o_{17} , o_{18} , o_{19} , o_{20} est supérieur à 0.5 ce qui indique que toutes ces ontologies sont liées à d'autres vocabulaires existants et contiennent un nombre élevé de classes externes et de références à des classes externes. Par exemple, le couplage de CIDOC-CRM est de 0,88, cette valeur indique que o_8 a un fort couplage. Par conséquent, les ontologies avec un fort couplage sont les plus difficiles à comprendre et à maintenir (Orme et al. 2006).

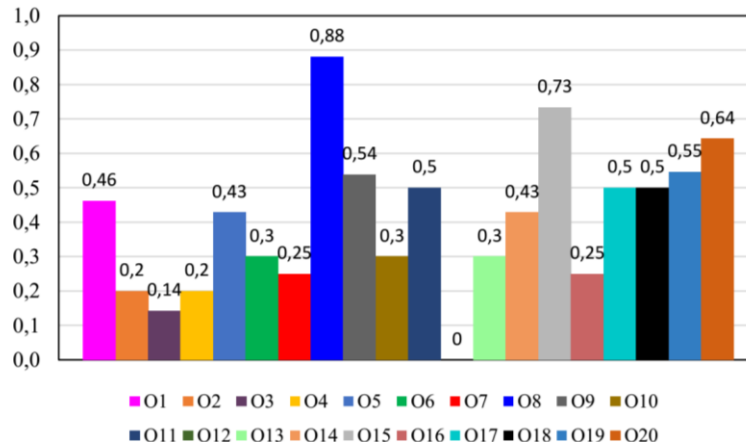


Figure 13: The coupling measure of the studied ontologies.

De manière générale, en utilisant ces métriques, nous constatons que la grande taille du vocabulaire, le plus grand nombre moyen de chemins par concept (ρ) et la longueur moyenne des chemins ($\bar{\Lambda}$) indiquent que les ontologies CH de l'ensemble de données sont très complexes. Par conséquent, il serait conseillé d'envisager la réutilisation et le partage de ces ontologies plutôt que d'essayer de construire des ontologies similaires à partir de zéro. Grâce à la métrique TIP, l'ingénieur en ontologie peut vérifier si la conception de l'ontologie suit de bons principes de classification. Grâce à la métrique de couplage, l'ingénieur en ontologie peut vérifier la relation entre l'ontologie locale et les ontologies externes. Par conséquent, l'analyse du TIP et du couplage a révélé que la majorité des ontologies CH étudiées peuvent être difficiles à maintenir.

On peut penser que le plus grand nombre de classes, de propriétés et d'axiomes est un point fort pour étudier la complexité de l'ontologie et on considère que plus le nombre de métriques primitives est grand, plus une ontologie est complexe. Cependant, nous soutiendrons qu'il est très difficile de mesurer la complexité d'une ontologie avec des métriques primitives. Prenons l'ontologie Fabio comme exemple. C'est l'une des plus grandes ontologies avec 261 classes. Une autre ontologie, la CIDOC-CRM (169 classes), contient un nombre de classes inférieur à celui de l'ontologie Fabio. Cependant, le résultat empirique montre que le CIDOC-CRM a un grand TIP par rapport au TIP de l'ontologie Fabio (CIDOC-CRM TIP = 2295, Fabio TIP = 1377). De plus, la métrique de couplage montre que le CIDOC-CRM a un fort couplage (Coupling=0.87) par rapport à l'ontologie FABIO (Coupling= 0.37). En d'autres termes, nous ne pouvons pas utiliser les métriques primitives pour mesurer l'aspect de la complexité de l'ontologie afin d'obtenir une compréhension plus complète de ces ontologies.

Conclusion

Dans ce chapitre, nous avons présenté deux types d'évaluation d'ontologie : la vérification et la validation. La tâche de vérification consiste à vérifier si l'ontologie est correctement construite. Par conséquent, la vérification est divisée en deux catégories : l'évaluation TBox et l'évaluation ABox. Les métriques les plus importantes pour vérifier la qualité de la TBox sont : la richesse des attributs, la richesse de l'héritage et la richesse des relations. Nous avons ensuite évalué l'ABox en utilisant la richesse des classes et la population moyenne.

D'autre part, la validation mesure la complexité d'une ontologie qui donne un aperçu aux développeurs et les aide à mieux comprendre, réutiliser, réduire les exigences de maintenance et intégrer les ontologies, ainsi qu'à aider les utilisateurs finaux à sélectionner l'ontologie qui répond le mieux à leurs besoins. Pour mesurer la complexité d'une ontologie, quatre métriques sont présentées : la taille du vocabulaire (SOV), l'impureté de l'arbre (TIP), le couplage, la longueur moyenne des chemins ($\bar{\Lambda}$), et le nombre moyen de chemins par concept (ρ).

Plusieurs expériences numériques sont réalisées, afin de valider le processus de mise en correspondance et de vérifier les ontologies existantes. Dans la première, nous avons sélectionné quatre bases de données du monde réel qui couvrent divers facteurs importants utilisés dans le processus de mappage tels que les tables, les vues, les colonnes, les clés étrangères (FK) et les clés primaires (PK). Ces facteurs représentent les métadonnées importantes extraites automatiquement de la BDR par nos algorithmes pour construire le fichier de mappage R2RML. À cet égard, nous avons constaté que notre processus de mappage préserve les requêtes et les informations. De plus, nous pouvons en déduire que notre processus de mapping nécessite des optimisations en termes de performance. Dans la seconde, nous analysons un ensemble de caractéristiques de complexité avancée de 20 ontologies du patrimoine culturel et leur impact sur l'évolution et la réutilisation de l'ontologie. Nous avons constaté que la grande taille du vocabulaire, le nombre moyen de chemins par concept (ρ) et la longueur moyenne des chemins ($\bar{\Lambda}$) indiquent que les ontologies CH de l'ensemble de données sont très complexes. Par conséquent, il serait conseillé d'envisager la réutilisation et le partage de ces ontologies plutôt que d'essayer de construire des ontologies similaires à partir de zéro. Grâce à la métrique TIP, l'ingénieur en ontologie peut vérifier si la conception de l'ontologie suit de bons principes de classification. Grâce à la métrique de couplage, l'ingénieur en ontologie peut vérifier la relation entre l'ontologie locale et les ontologies externes. Par conséquent, l'analyse du TIP et du couplage a révélé que la majorité des ontologies CH étudiées peuvent être difficiles à maintenir. De plus, nous avons prouvé qu'il est très difficile de mesurer la complexité d'une ontologie en se basant sur les métriques primitives telles que le nombre de classes, le nombre de propriétés et le nombre d'axiomes.

Enfin, le processus de développement d'une ontologie est obligatoirement suivi du processus d'évaluation avant de publier cette ontologie et de donner aux utilisateurs la possibilité de l'utiliser

dans des applications. Dans ce contexte, le point suivant que nous devons aborder est le stockage de l'ontologie. Nous devons stocker l'ontologie de manière à pouvoir la faire évoluer au fil du temps (par exemple en ajoutant de nouveaux types de connaissances) et accéder aux connaissances encodées dans l'ontologie de manière efficace. Cette tâche correspond à la couche de stockage des connaissances dans l'architecture du graphe de connaissances.

Chapter3. Les systèmes de stockage RDF

Dans ce chapitre, nous présentons trois nouvelles implémentations de méthodes non natives pour le stockage de données RDF. Ces méthodes, nommées RDFSPO, RDFPC et RDFVP, sont basées respectivement sur les approches de table d'énoncés, de table de propriétés et de partitionnement vertical. Nous considérons comme importante la question de savoir comment sélectionner la stratégie la plus pertinente pour stocker les données RDF en fonction des caractéristiques de données. Pour cela, nous étudions l'équilibre entre deux mesures de performance, notamment le temps de chargement et le temps de réponse aux requêtes. Dans ce contexte, nous fournissons une étude comparative empirique entre, d'une part, les trois méthodes proposées et, d'autre part, les méthodes proposées par rapport aux méthodes existantes en utilisant divers ensembles de données disponibles publiquement. Enfin, afin de mieux évaluer où les différences statistiquement significatives apparaissent entre les méthodes étudiées, nous avons effectué une analyse statistique, basée sur le test non-paramétrique de Friedman suivi d'un test post-hoc de Nemenyi.

1. Importance du référentiel sémantique

1.1. Définition: Référentiel Sémantique

Un entrepôt sémantique est un système de gestion de base de données qui permet de stocker, d'interroger et de gérer des données structurées. En effet, le terme de référentiel sémantique n'est pas encore largement adopté et il est souvent appelé base de données de graphes sémantiques, raisonneur, serveur d'ontologie, magasin sémantique, magasin de métadonnées, base de données RDF et triplestore RDF (Modoni et al. 2014). Par rapport aux SGBD traditionnels tels que les bases de données relationnelles, le principal avantage des référentiels sémantiques est l'utilisation du paradigme de données sémantiques, appelé modèle de données RDF (Thakker et al. 2010). En fait, le modèle de données RDF permet de changer le schéma de données à la volée sans interférer avec les données, de découvrir de nouveaux faits et de construire de nouvelles données basées sur des règles sémantiques en utilisant la capacité d'inférence, et d'intégrer de manière transparente des données provenant de diverses sources (Gayo et al. 2017b).

Les avantages de l'utilisation de référentiels sémantiques sont apparus dans leur capacité à ingérer des données diverses et à fournir une flexibilité en ce qui concerne les changements de schémas et les mappings. Ces bases de données (également connues sous le nom de RDF, OWL ou Graph Databases) sont désormais largement utilisées pour gérer des données non structurées et structurées dans les domaines des médias et de l'édition, des sciences de la vie et des services financiers.

1.2. Pourquoi Référentiel Sémantique

Contrairement aux bases de données NoSQL orientées documents et aux moteurs de recherche en texte intégral, les triplestores permettent des requêtes structurées complètes. Ils sont uniques par leur capacité à interpréter les données par le raisonnement et à découvrir de nouveaux faits et relations. Pour ce faire, les triplestores s'appuient sur des schémas sémantiques (appelés ontologies) qui fournissent des définitions formelles de la signification des types d'objets et de relations. Leur capacité à interpréter les données permet aux triplestores de fournir des réponses riches et profondes même lorsque les requêtes ignorent la syntaxe dans laquelle les données sont ingérées. Obtenir des données diverses à partir de sources multiples demande moins de travail. Le stockage et la maintenance de dizaines de milliards de données sur un seul serveur de base permettent de réduire les coûts de matériel. Les options de cloud computing peu coûteuses, entièrement hébergées et à la demande, constituent une alternative peu onéreuse aux solutions d'entreprise.

Pour net pourquoi les triplestores sont importants, voici quelques résons :

Flexibilité des schémas - Les Triplestores sont flexibles à bien des égards. Ils ne nécessitent pas le développement de schémas complexes comme c'est le cas pour les bases de données relationnelles. Les données peuvent donc être facilement chargées dans les triplestores. Il existe des outils qui peuvent être utilisés pour traduire vos données au format RDF si nécessaire. Comme les triples et les URI présentent tous les mêmes formats, vous pouvez charger des quantités massives de triples sur différents sujets dans le même triplestore. Il s'agit d'un très grand réservoir de connaissances qui peut croître en permanence.

Facilité d'interrogation - Les triplestores permettent d'interroger des types de relations générales ou même non spécifiées, libérant ainsi les développeurs d'applications de la nécessité de connaître les spécificités de chaque source de données et de modifier les requêtes chaque fois que l'une des sources change. Vous pouvez avoir autant de langues de faits stockées dans le même triplestore que vous le souhaitez. Certains fournisseurs ont trouvé le moyen d'interroger simultanément le texte intégral et les données structurées. Ces requêtes hybrides offrent une flexibilité encore plus grande et des informations supplémentaires.

Normes - Alors que les bases de données NoSQL sont généralement toutes différentes les unes des autres, tous les triplestores partagent les mêmes normes, ce qui permet aux organisations d'échanger un triplestore contre un autre. Le déplacement des données d'un triplestore à un autre est également une tâche facile.

Provenance - Le suivi de l'origine des données a de nombreux cas d'utilisation importants. Les magasins tripartites peuvent maintenir ce niveau de détail. Les utilisateurs connaissent non seulement les faits sémantiques, mais aussi les sources de données, les dates, les niveaux de confiance et d'autres métadonnées sur les triplets. Vous recherchez des faits sur la ville de New

York fournis au début du siècle dernier par des sources spécifiques ? Ce n'est pas un gros problème pour un triplestore.

2. La classification des systems de stockage RDF

Parmi la pléthore de systèmes existants pour le stockage du graphe de connaissances, nous pouvons distinguer les solutions qui implémentent leur propre backend de stockage, dénommées systèmes natifs, et celles qui utilisent un système de gestion de base de données existant, dénommées systèmes non-natifs. Une classification des techniques de stockage des données RDF est proposée dans la figure 19.

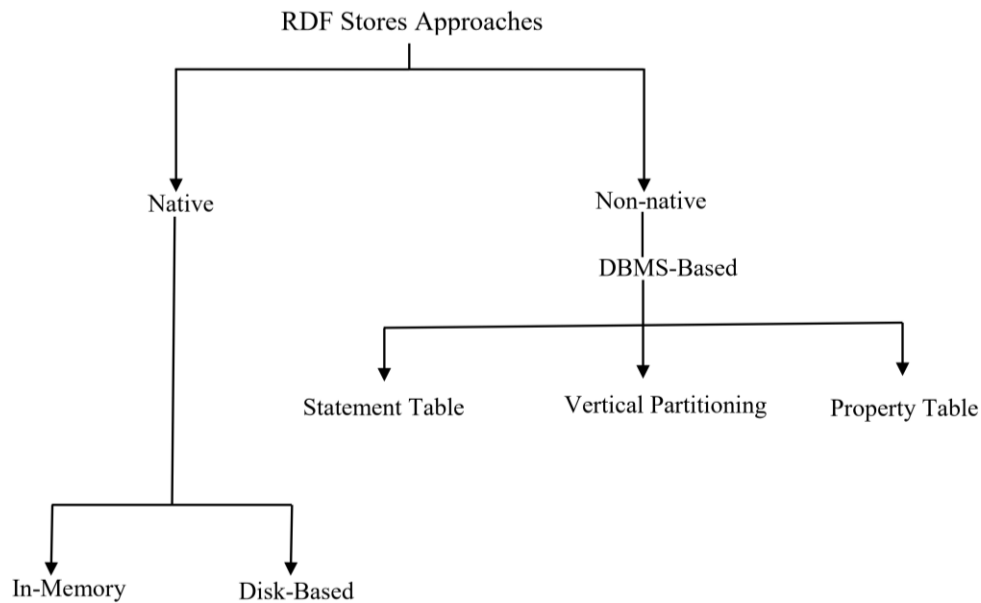


Figure 14: Classification of the RDF storage systems

2.1. Systèmes Native

L'approche native fournit un moyen de stocker les données RDF plus près de leur modèle de données, en évitant la mise en correspondance avec les entités d'un SGBD, comme les relations. Elle utilise la nature triples des données RDF comme un atout et permet de s'attaquer aux spécificités de son approche graphique, telles que la capacité à gérer la sparsité des données et l'aspect dynamique de son schéma. Ces systèmes peuvent être classés en deux grandes catégories : les systèmes sur disque, c'est-à-dire persistants, et les systèmes en mémoire principale, c'est-à-dire volatils.

Les systèmes natifs désignent les systèmes qui ne tirent pas profit des systèmes existants pour stocker les données RDF. Ils sont construits à partir de zéro sur la base d'une technique d'indexation spécifique au modèle de données RDF (Pan et al. 2018). En fait, les systèmes natifs offrent une plus grande flexibilité que les bases de données traditionnelles et réduisent le temps de chargement

des données. En outre, les systèmes natifs offrent de nombreuses autres fonctions traditionnelles des bases de données, telles que le traitement des transactions, le contrôle d'accès, la journalisation et la récupération des données. Plus précisément, ces systèmes peuvent être largement classés en systèmes persistants sur disque et en systèmes à mémoire principale (Pan et al. 2018).

Le stockage persistant sur disque est un moyen de stocker les données RDF de manière permanente sur un système de fichiers en utilisant les techniques d'indexation les plus influentes, telles que l'arbre B+ (Singh et al. 2018), AVL (Wood et al. 2005) et B-tree (Harth and Decker 2005). Parmi les solutions existantes, nous pouvons citer (Broekstra et al. 2002; Aasman 2006; Harth et al. 2007; Kolas et al. 2009; Tran et al. 2009; Chen and Reformat 2014). Il est important de noter que la lecture et l'écriture sur les disques ralentissent le processus de recherche à un niveau inacceptable et constituent un important goulot d'étranglement en termes de performances (Curé and Blin 2014).

Pour surmonter ce problème, les solutions in-memory ont été utilisées. Le stockage en mémoire alloue une certaine quantité de la mémoire principale (RAM) pour stocker l'ensemble des données RDF. Lorsque l'on travaille sur des données RDF stockées en mémoire principale, certains des facteurs les plus importants à prendre en compte sont le chargement et l'analyse du fichier RDF (Curé and Blin 2014). Par conséquent, le stockage RDF qui utilise l'approche en mémoire doit avoir une représentation des données efficace en termes de mémoire qui laisse suffisamment d'espace pour le fonctionnement des algorithmes de recherche. Les travaux suivants entrent dans cette catégorie (Broekstra et al. 2002; Reggiori et al. 2003; Wilkinson et al. 2003; Guha 2004; Janik and Kochut 2005; Weiss et al. 2008; Atré et al. 2009; Fletcher and Beck 2009).

2.2. Systèmes Non-Native

Non natifs se réfèrent particulièrement aux systèmes qui utilisent les systèmes de gestion de bases de données relationnelles (SGBDR) ou d'autres systèmes connexes pour stocker les données RDF de manière permanente. Actuellement, le SGBDR est largement considéré comme l'endroit le plus performant pour les données RDF persistantes en raison des grands efforts réalisés pour développer des solutions qui rendent le stockage des données RDF efficace, évolutif et robuste (Ma et al. 2016). Afin de discuter des magasins RDF sur SGBDR, la première question à couvrir est de savoir comment mapper les données RDF sur des tables relationnelles. À cet égard, il existe trois stratégies de stockage qui sont : La table des déclarations, la table des propriétés et le partitionnement vertical (Curé and Blin 2014).

La table Statement (Karvinen et al. 2019) est la manière la plus simple de mapper des données RDF à une base de données relationnelle. Comme le montre le tableau 16, elle consiste à créer une table à trois colonnes (sujet, prédicat, objet), où chaque ligne correspond séparément à une déclaration RDF. En fait, la façon dont toutes les données sont combinées en une seule grande table pose le problème de la faible efficacité de la requête, puisqu'une simple requête SELECT nécessite un grand nombre d'auto-joints. Plus précisément, si le nombre de déclarations RDF augmente, le

temps de réponse de la requête augmentera avec l'incrémentation du nombre d'auto-joints. Afin d'améliorer l'efficacité des requêtes, les techniques d'indexation sont alors ajoutées pour chacune des colonnes afin de réduire le coût des requêtes auto-jointes (Karvinen et al. 2019). Cependant, le stockage des triplets RDF dans une seule table rend l'exécution des requêtes très lente et peut dépasser la taille de la mémoire principale, comme indiqué dans De nombreux premiers magasins RDF utilisent l'approche des tables de déclaration, comme(Beckett 2002; Harris and Gibbins 2003; Guha 2004; Erling and Mikhailov 2009; McGlothlin and Khan 2009a, b).

Subject	Predicate	Object
ID1	type	Book
ID1	title	"A Developer's Guide to the Semantic Web "
ID1	author	"Liyan Yu"
ID1	date	"2014"
ID2	Type	Movie
ID2	Title	"A Beatiful Mind"
ID2	Actor	"Russel Crowe"
ID2	date	"2001"
ID2	Language	"English"
ID3	Type	Book
ID3	title	"Unlimited Memory"
ID3	language	"English"
ID4	Type	Journal
ID4	Title	"Journal On Data Semantics"
ID5	Title	"One Man's Dream"
ID5	Date	"1993"
ID5	Type	Movie
ID6	type	Book
ID6	date	"2015"

Table 11: Statement table approach.

Property table (PT) (Abadi et al. 2007a; Karvinen et al. 2019) a été proposée plus tard et peut être classée en deux types : la table de propriétés groupées (CP) et la table de classes de propriétés (PC). La première contient des clusters de propriétés qui tendent à décrire le même sujet (tableau 17). La seconde exploite le prédicat "rdf:type" pour regrouper des ensembles similaires de sujets dans la même table (Tableau 18). En général, l'idée principale est de découvrir des groupes de sujets apparaissant souvent avec le même ensemble de propriétés. Cette approche a été présentée dans plusieurs travaux comme (Alexaki et al. 2001; Broekstra et al. 2002; Wilkinson et al. 2003; Harris et al. 2009a). En fait, la conséquence immédiate de l'application de PT est que les requêtes SPARQL complexes peuvent être récupérées sans un coûteux self-joins. Cependant, comme indiqué dans(Abadi et al. 2007a) l'approche PT présente trois inconvénients majeurs. Le premier est le problème de la génération de nombreuses valeurs nulles, ce qui entraîne une surcharge de performance substantielle. Le deuxième est que le PT ne peut pas gérer les propriétés multivaluées. Le troisième fait référence aux requêtes complexes qui prouvent que le PT est encore coûteux, car la plupart de ces requêtes nécessitent des clauses d'union et des jointures pour collecter les données de plusieurs tables. À cet égard, une solution alternative a été proposée, à savoir l'approche de partitionnement vertical.

Subject	Type	Title	Date
ID1	Book	"A Developer's Guide to the Semantic Web "	"2014"
ID2	Movie	"A Beatiful Mind"	"2001"
ID3	Book	"Unlimited Memory"	NULL
ID4	Journal	"Journal On Data Semantics"	NULL
ID5	Movie	"One Man's Dream"	"1993"
ID6	Book	NULL	"2015"

Table 12: Clustered property table (CP).

Class: Book

Subject	Title	Author	Date
ID1	"A Developer's Guide to the Semantic Web "	"Liyan Yu"	"2014"
ID3	"Unlimited Memory"	NULL	NULL
ID6	NULL	NULL	"2015"

Class: Movie

Subject	Title	Actor	Date
ID2	"A Beatiful Mind"	"Russel Crowe"	"2001"
ID5	"One Man's Dream"	NULL	"1993"

Table 13: Property Class approach (PC).

Le partitionnement vertical (VP)(Abadi et al. 2007a; Karvinen et al. 2019) fait référence à la division verticale de l'énoncé RDF en fonction du prédicat. Comme le montre le tableau 19, les triplets RDF sont divisés en n tables à deux colonnes, où n est le nombre de prédicats uniques dans l'ensemble de données RDF. Dans chacun de ces tableaux, la première colonne comprend les sujets définis par le prédicat et la deuxième colonne comprend les valeurs des objets de ces sujets. Il est important de mentionner que le nom du prédicat peut être utilisé comme nom de table. Par rapport à l'approche PT, l'approche VP fournit un support pour les attributs à valeurs multiples, et les valeurs nulles sont simplement omises de la table. En outre, pour une requête donnée, seule la table correspondant aux propriétés impliquées dans cette requête doit être lue, et aucun algorithme de clustering n'est nécessaire pour diviser les triples RDF en deux colonnes. L'approche VP est utilisée dans plusieurs travaux comme(Abadi et al. 2007a; Faye et al. 2012; MahmoudiNasab and Sakr 2012).

Title		Type		Date		Language	
ID1	"A Developer's Guide to the Semantic Web "	ID1	Book	ID1	"2014"	ID2	"English"
ID2	"A Beatiful Mind"	ID2	Movie	ID2	"2001"	ID3	"English"
ID3	"Unlimited Memory"	ID3	Book	ID5	"1993"	Author	
ID4	"Journal On Data Semantics"	ID4	Music	ID6	"2015"		
ID5	"One Man's Dream"	ID5	Movie	Actor		ID1	"Liyan Yu"
		ID6	Book				
				ID2	"Russel Crowe"		

Table 14: Vertical partitioning approach (VP)

3. Implémentation proposées

Dans cette section, nous proposons une implémentation des trois approches présentées dans la section précédente. Ces implémentations sont nommées RDFSPO, RDFPC, et RDFVP, qui sont

basées respectivement sur les approches de table d'énoncés, de tables de propriétés et de partitionnement vertical (Abadi et al. 2007b).

3.1. RDF Subject-Predicate-Object method (RDFSPO)

La méthode RDFSPO est un magasin non natif qui consiste à stocker les données RDF en utilisant le SGBDR comme backend. De manière concise, l'algorithme principal de mise en œuvre de la méthode RDFSPO est décrit dans l'algorithme 1. En général, la méthode RDFSPO mappe les données RDF directement sur une table à trois colonnes SPO (subject-predicate-object). Il est important de mentionner que nos algorithmes de mappage des données RDF sur la base de données relationnelle sont automatiques et génériques.

Algorithm 1: RDF Subject-Predicate-Object Method

Input: RDF dataset

Output: RDF dataset stored in RDB.

Var:

```

    iter: stmtIterator; model: Model; in: InputStream; t: Triple; stmt: Statement;
    subject: Resource; predicate: Property; object: RDFNode;
Begin
1:   DBUTILS.createSPOTable();
2:   model ← ModelFactory.createDefaultModel();
3:   in ← FileManager.get().open(inputFileName);
4:   if (in is NULL) then
5:     ThrowExcpetion();
6:   Else
7:     Model.read(in, " ");
8:   End if
9:   iter ← model.displayAllStatment();
10:  while (iter.hasNext())
11:    stmt ← iter.nextStatment();
12:    subject ← stmt.getSubject();
13:    predicate ← stmt.getPredicate();
14:    object ← stmt.getObject();
15:    if (object ∈ Resource) then
16:      t.setObjet(Object.toString());
17:    else
18:      t.setObject(object.toString());
19:    end if
20:  end while
21:  DBUTILS.insertIntoSPOTable(t);
22: END

```

3.2. RDF Property Clustering method (RDFPC)

Comme nous l'avons mentionné dans la section précédente, l'approche des tableaux d'états présente plusieurs inconvénients évidents. Plus précisément, la méthode RDFSPO génère une grande quantité de réplication inutile d'informations qui apparaissent dans plusieurs lignes. A cet égard, afin de contourner toutes les limitations introduites par la méthode RDFSPO, une autre

implémentation est proposée. L'algorithme proposé pour cette implémentation est donné dans l'algorithme 2

Algorithm 2: RDF Clustered Property Method

Input: RDF dataset

Output: RDF dataset stored in RDB.

Var:

properties: Hashset<Property>; list: List<RDFNode>; stmt: Statement; subject: Resource;
predicate, pre: Property; object, obj: RDFNode; s, str: table

Begin

```

1:  s←node.toString().split("/");
2:  last←s.length-1;
3:  if(s[last].contains("#") then
4:    scriptSQL←createTable(str[1])
5:  else
6:    scriptSQL←createTable(str[last])
7:  end if
8:  stmt←model.displayAllStatment(subject,null,null)
9:  while(stmt.hasNext())
10:   if(s[last].contains("#")) then
11:    str←s[last].split("#");
12:    insertScriptSQL←insertPC(str[1]);
13:   else
14:    insertScriptSQL←insertPC(str[0]);
15:   end if
16:   stmt1←model.displayAllStatment(subject,null,nul);
17:   while(stmt1.hasNext())
18:    properties.add(pre);
19:    list.add(obj);
20:   end while
21:   for(Property pr: properties)
22:    if(pr.getLocalName().toString().equals("type"));
23:    scriptSQL←insertScriptSQL.replace("[", " ");
24:    scriptSQL←insertScriptSQL.replace("]", " ");
25:   end if
26: end For
27:   file←writeToFile(insertScriptSQL)
28: end while
29: DBUTILS.insertIntoClusterdTable(file)
30: End

```

3.3. RDF Vertical Partitioning method (RDFVP)

La méthode RDFVP est une alternative à l'implémentation RDFPC, où nous pouvons omettre toutes les limitations de la méthode RDFPC (basée sur l'approche de la table de propriété) et accélérer les requêtes sur le jeu de données RDF. La mise en œuvre de la méthode RDFVP est décrite dans l'algorithme 3.

Algorithm 3: RDF Vertical Partitioning Method

Input: RDF dataset

Output: RDF dataset stored in RDB.

Var:

iter: stmtIterator; model: Model; in:InputStream;

```

t: Triple;stmt: Statement;subject: Resource;
predicate: Property;object: RDFNode;listOfProperties: List
Begin
1:  model←ModelFactory.createDefaultModel();
2:  in←FileManager.get().open(inputFileName);
3:  if(in is NULL) then
4:    ThrowException()
5:  else
6:    Model.read(in," ");
7:  end if
8:  listOfProperties←listOfPredicate();
9:  for each pr in listOfProperties
10:    tblName←pr.getLocalName().toString();
11:    DBUtils.createTable(tblName);
12:  end for
13:  stmt←DisplayAllStatement(null,predicate,null)
14:  while(stmt.hasNext())
15:    state←stmt.nextStatment();
16:    subject←state.getSubject();
17:    object←state.getObject();
18:    t.setSubject(subject);
19:    t.setObject(object);
20:    DBUtils.isertIntoTable(tblName,t);
21  end while
22  DBUtils.createNewTable(tblName);
23  tstListStatmentByProperty(predicate);
24  end

```

4. Configuration des expériences

Un ensemble d'expériences appropriées a été organisé afin d'étudier les caractéristiques de performance des systèmes de gestion de données RDF, analysés dans les sections précédentes. Pour cette raison, un ensemble de jeux de données bien connus de la littérature sont sélectionnés. Les informations détaillées de ces jeux de données sont résumées dans le tableau 20. Comme preuve de concept, nous avons exécuté notre étude expérimentale contre quatre magasins RDF populaires, y compris Jena TDB(Curé and Blin 2014), Jena SDB(Curé and Blin 2014), RDF4jM(Broekstra et al. 2002), and RDF4jD(Broekstra et al. 2002) en plus de trois méthodes proposées qui utilisent le backend MYSQL, qui sont RDFSPO, RDFPC et RDFVP. Dans ce contexte, nous avons collecté 17 jeux de données pour tester la performance dimensionnelle de ces systèmes. Enfin, il est important de noter que toutes les simulations expérimentales ont été réalisées sur un ordinateur personnel sous Windows 10, avec un processeur intel core i7 2,70 GHZ et 16 Go de RAM.

Datasets	#Triples	#Subjects	#Predicates	#Objects
allSWGroups ¹	345	56	19	165
Wikimovies ⁴	505	157	22	121
NATS ²	4036	266	33	568
Tissues ³	4113	1426	7	2663
Locations ⁶	5374	949	17	2885
VSRR ⁴	6221	500	13	1387
AQMNEHTN ⁵	8000	500	16	1456
CDI ⁶	13462	1000	29	1493
DHDS ⁸	14802	1000	30	2477
DASH ⁸	16152	1000	33	4635
Journals ⁶	37581	4 473	12	29358
Diseases ⁶	69487	18061	8	36185
Enzyme ⁶	84398	14 633	14	47 391
GeneSymbol ⁶	119485	17077	7	85345
Protein ⁵	160537	14635	8	117034
swdf ⁷	242249	23308	171	76529
Geo_coordinates_en ⁸	1569180	496990	4	1045824

Table 15: The basic Statistics of Datasets

4.1. Description des systems RDF existants

Cette sous-section est consacrée à la description des quatre systèmes existants que nous avons utilisés pour démontrer l'efficacité des méthodes que nous proposons. Ces systèmes sont : Jena TDB, Jena SDB, RDF4J basé sur la mémoire principale (RDF4JM), et RDF4J basé sur le disque (RDF4JD). Le tableau 21 donne un aperçu des systèmes de stockage existants que nous avons utilisés dans cette étude.

¹ <https://www.wikidata.org>

² https://www.cdc.gov/tobacco/data_statistics/surveys/nats/index.htm

³ <https://www.uniprot.org/>

⁴ <https://www.cdc.gov/nchs/nvss/vsrr/drug-overdose-data.htm>

⁵ <https://healthdata.gov/search/type/dataset>

⁶ <https://www.cdc.gov/cdi/index.html>

⁷ <https://lod-cloud.net/dataset/semantic-web-dog-food>

⁸ <https://wiki.dbpedia.org/Datasets>

RDF storage system	Approach	Category	Reasoning Strategy	Query Language
Jena SDB	Statement table	Non-native	Backward chaining	SPARQL
Jena TDB	Disk-based	Native	Backward chaining	SPARQL
RDF4JD	Disk-based	Native	Forward chaining	SPARQL
RDF4JM	Main memory-based	Native	Forward chaining	SPARQL

Table 16: Characteristics of proposed RDF storage systems

4.2. Implémentation des requêtes

Pour tester le temps de réponse des requêtes dans les différents systèmes de stockage RDF, nous avons utilisé 12 requêtes que nous détaillerons plus tard. Il y a deux façons différentes de créer une requête. La première méthode consiste à concevoir une requête basée sur certaines caractéristiques et à évaluer ainsi le fonctionnement de ces caractéristiques. La seconde méthode consiste à concevoir une requête en se basant davantage sur des cas d'utilisation réels. Pour notre étude expérimentale, nous adoptons certaines requêtes du SP²Bench(Schmidt et al. 2009) qui sont basées sur la deuxième méthode, combinées avec d'autres requêtes construites avec la première méthode. Le tableau 22 résume toutes les requêtes testées dans cette étude expérimentale. En général, ces requêtes varient également en fonction de leurs caractéristiques comme le montre le tableau 23.

1. **Q1.** Return the list of authors and titles of papers that have a rdf:type value proceeding.
2. **Q2.** Select all proceedings papers with property swrc:pages or swrc:month or swrc:isbn.
3. **Q3.** Extract all proceedings papers with properties dc:creator, swrc:booktitle, swrc:isPartOf, rdfs:seeAlso, dc:title, swrc:homepage,dc:subject and optionally swrc:abstract, considering their values.
4. **Q4.** Return all proceedings papers that they are accepted in iswc conference 2008.
5. **Q5.** Select all distinct pairs of paper author names for authors that have published in the same journal.
6. **Q6.** Return all full papers and short papers accepted in the rule-ML conference 2011.
7. **Q7.** Extract all proceedings papers with the machine learning topic.
8. **Q8.** Extract 10 resources that are somehow related to each other.
9. **Q9.** Return All the proceedings papers accepted in all conferences described in swdf dataset with the semantic web or machine learning topics.
10. **Q10.** Return all subjects that stand in any relation to person "Annabel Bourde".
11. **Q11.** Compute authors that have published with Annabel Bourde or with an author that has published with Annabel Bourde.
12. **Q12.** Extract all the name of persons and their papers including the type and the subject.

Table 17: The tested queries implemented in this experimental study.

characteristic	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Simple Query	x	-	-	-	-	-	-	x	-	x	-	-

filters	-	×	-	-	×		-	-	-	-	×	-
More than 6 patterns	-	-	×	-	×	×	-	-	×	-	-	×
Optional operator	-	-	×	-	-	-	-	-	-	-	-	-
Limit Modifier	-	-	-	-	-	-	-	×	-	-	-	-
Offset Modifier	-	-	-	-	-	-	-	×	-	-	-	-
Union Operator	-	-	-	-	-	×	-	-	×	-	×	-
Distinct Operator	-	-	-	×	×	-	×	-	×	-	×	×
Order By	-	-	×	-	-	-	-	-	-	-	-	-
And operator	×	×	×	×	×	×	×	×	×	-	×	×
Or operator	-	×	-	-	-	-	-	-	-	-	-	-
Type of Query	SQ	SQ	SQ	SQ	TQ	ST	TQ	-	TQ	-	CQ	CQ

Table 18: Characteristics of tested queries. SQ= Start Query, TQ=Tree Query and CQ= Chain Query.

Comme le montre le tableau 4, les requêtes testées entrent dans l'une des trois catégories suivantes : (1) Requête en étoile : c'est le type le plus fréquemment utilisé, il consiste uniquement en des jointures sujet-sujet où une variable de jointure est représentée par la pièce sujet de tous les modèles triples impliqués dans la requête ; (2) Requête en chaîne : comprend des jointures sujet-objet où les modèles triples sont consécutivement liés comme une chaîne. (3) Requête arborescente : comprend des jointures sujet-sujet et des jointures sujet-objet(Hassan and Bansal 2018).

5. Résultats expérimentaux et discussion

L'objectif principal de ce travail est d'évaluer les différents systèmes de stockage de données RDF, qu'ils soient natifs ou non-natifs, afin de choisir la stratégie la plus appropriée pour stocker les données RDF en fonction de caractéristiques spécifiques. Dans ce contexte, les résultats expérimentaux sont discutés comme suit. Premièrement, l'évaluation de nos implémentations proposées en termes de temps de chargement et de temps de réponse aux requêtes. Deuxièmement, l'évaluation des systèmes existants toujours en termes de temps de chargement et de temps de réponse aux requêtes. Enfin, la comparaison entre les systèmes proposés et existants. Plus précisément, la mesure du temps de chargement est répétée 10 fois et le temps CPU moyen écoulé est calculé, tandis que le temps de réponse aux requêtes est calculé en prenant la moyenne du temps de réponse de l'exécution de chaque requête dix fois consécutivement. Dans notre étude, nous avons utilisé 12 requêtes qui ont été définies pour répondre aux questions de la vie réelle. Le jeu de données utilisé dans notre analyse est le SWDF, qui contient 242249 triplets.

5.1. L'évaluation des systems proposés

Dans la première partie de cette sous-section, nous présentons une évaluation des trois systèmes proposés, à savoir RDFSPO, RDFPC et RDFVP en termes de temps de chargement et de temps de réponse aux requêtes.

5.1.1. Mesure du temps de chargement

Si l'on considère le résultat présenté à la figure 20 (a), on voit clairement que le RDFVP est considérablement plus performant que le RDFPC et le RDFSPO. D'une part, l'idée fondamentale derrière le VP est le partitionnement des données en utilisant un modèle de stockage entièrement décomposé (Copeland and Khoshafian 1985; Corwin et al. 2007). Puisque les données proviennent de différentes tables de la même base de données, il est plus facile de les gérer qu'une situation dans laquelle les mêmes ensembles de données sont stockés dans une seule table comme dans le cas du RDFSPO.

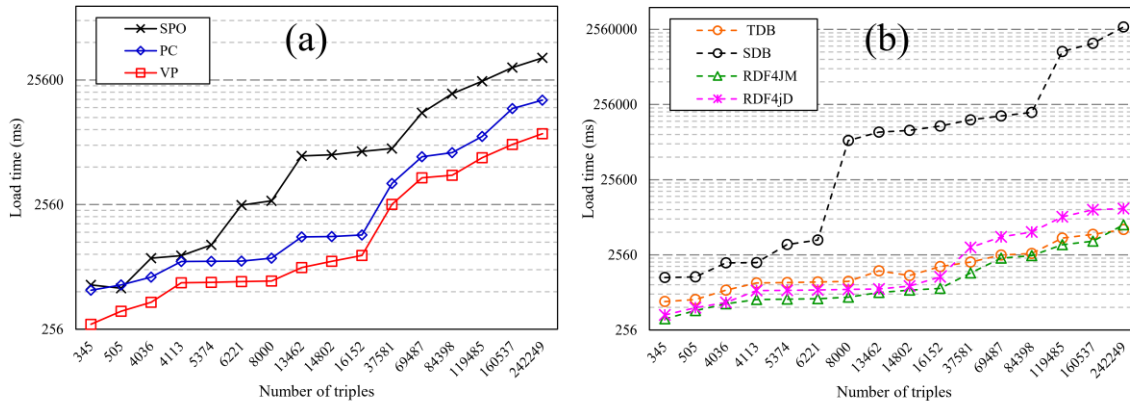


Figure 15 Load Time. Figure (a) shows the load time of the proposed methods. Figure(b) shows the existing methods:

D'autre part, la raison pour laquelle le RDFVP obtient de meilleures performances que le RDFPC est associée au processus utilisé par le RDFPC pour charger les données. En effet, le RDFPC utilise une stratégie de regroupement basée sur la propriété "rdf:type" pour stocker les données RDF dans une base de données relationnelle. Cela signifie qu'avant de charger les données, le RDFPC prend en considération le regroupement des données RDF basé sur la propriété "rdf:type". Par conséquent, le temps de chargement est calculé comme la somme du temps de regroupement des données RDF et du temps de chargement.

5.1.2. Temps de réponse des requêtes

En considérant les résultats présentés dans la figure 23, RDFSPO est très lent à répondre à toutes les requêtes par rapport à RDFPC et RDFVP. Ceci est essentiellement dû à l'utilisation d'une seule table pour le stockage de tous les triplets. En effet, lorsque le nombre de triplets augmente, cette table unique peut dépasser la taille de la mémoire principale (Abadi et al. 2007a). De plus, les

requêtes complexes avec des motifs multiples nécessitent de nombreuses auto-jointes, ce qui entraîne des performances médiocres. Une autre observation intéressante que nous pouvons faire sur le RDFSPO est qu'il utilise la recherche par balayage de table pour trouver les enregistrements appropriés, ce qui conduit à ralentir le temps d'exécution de la requête (Abadi et al. 2007a).

Comme le montre la figure 23, la méthode RDFPC est plus performante que les autres méthodes testées. En même temps, elle suit de près la méthode RDFVP proposée, qui donne les meilleurs résultats pour toutes les requêtes. La conséquence immédiate de la méthode RDFPC est qu'elle peut éviter le nombre excessif d'auto-joints générés par la méthode RDFSPO. Plus précisément, alors que la méthode RDFPC améliore les performances en réduisant le nombre d'auto-joints et le prédicat `rdf:type`, elle introduit de la complexité en stockant des informations inutiles comme les valeurs nulles, et peut causer la perte d'informations en ce qui concerne la gestion de la relation many-to-many. Par conséquent, nous pouvons clairement voir que la méthode RDFVP est plus performante que la méthode RDFPC. Dans l'ensemble, la méthode RDFVP permet une amélioration significative des performances en surmontant les limites rencontrées dans la méthode RDFPC. Par conséquent, il pourrait être présenté comme la meilleure alternative prometteuse aux modèles existants pour la récupération de données dans des référentiels.

5.2. L'évaluation des systèmes RDF existants

De manière similaire à la section précédente, dans cette deuxième section, nous allons comparer le temps de chargement et le temps de réponse aux requêtes pour le stockage des données RDF en utilisant les systèmes existants, à savoir TDB, SDB, RDF4JM et RDF4JD.

5.2.1. Mesure du temps de chargement

En analysant les résultats de la Figure 20 (b), il s'avère que SDB a montré une mauvaise performance de charge tandis que RDF4JM est rapide pour les petits ensembles de données par rapport à TDB et RDF4JD. Nous avons testé RDF4JM et RDF4JD sur des ensembles de données à plus grande échelle. Nous observons qu'ils n'ont pas bien fonctionné lors du chargement des données, comme on peut le voir sur la figure 8 (b). Par exemple, pour le chargement de 242249 triples, RDFJM, RDF4JD, et TDB ont pris respectivement 6423,1 ms, 10576,4 ms, et 5551,87 ms. Cela conduit à une remarque intéressante qui peut être faite sur les performances de ces référentiels en termes d'évolutivité : Le système sur disque (RDF4JD) et le système en mémoire (RDF4JM) s'adaptent bien aux petits ensembles de données, tandis que le système persistant (TDB) peut s'adapter aussi bien aux petits qu'aux grands ensembles de données.

En fait, les performances obtenues de ces systèmes peuvent être justifiées par la raison suivante : L'applicabilité des différentes stratégies de raisonnement. Plus précisément, les systèmes RDF4JD et RDF4JM utilisent la stratégie de chaînage avant. Cela signifie que le chargement des données est plus lent car le référentiel étend la fermeture inférée après chaque transaction. En d'autres termes, pendant le chargement, les données inférées doivent être stockées.

D'autre part, la TDB utilise la stratégie de chaînage en amont, et par conséquent, son chargement des données est plus rapide que celui des autres référentiels utilisant le chaînage en aval, car moins de temps est nécessaire pour le calcul et la maintenance des données inférées (Domingue et al. 2011b). Enfin, nous pouvons remarquer que même si le système SDB utilise également la stratégie de chaînage arrière, il présente un temps de calcul plus élevé, contrairement à la méthode TDB. En effet, cette augmentation du temps de chargement peut être liée au fait que la SDB implique des étapes supplémentaires, à savoir, le chargement des données dans la machine virtuelle java et ensuite sur la base de données pour le stockage des données (Curé and Blin 2014).

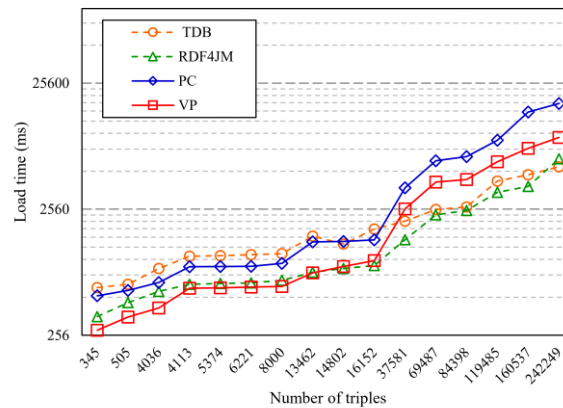


Figure 16: load time of the proposed methods against the existing ones.

5.2.2. Le temps de réponses des requêtes

En ce qui concerne le temps de réponse aux requêtes des méthodes existantes (Figure 23), nous pouvons observer que RDF4JD et RDF4JM donnent de meilleures performances par rapport à SDB et TDB. Cela est dû à la stratégie d'inférence utilisée pour récupérer les données. Plus précisément, RDF4JM et RDF4JD utilisent la stratégie de chaînage avant, ce qui signifie qu'aucune déduction ni vérification de la satisfiabilité n'est nécessaire lorsque nous interrogeons le référentiel (Domingue et al. 2011b). En revanche, les méthodes TDB et SDB utilisent le chaînage en amont. Dans ce cas, le temps de réponse aux requêtes est plus lent car une réécriture extensive des requêtes (expansion et reformulation) doit être effectuée (Domingue et al. 2011b). En fait, SDB n'est pas en mesure de rivaliser avec les autres systèmes existants et affiche des performances médiocres. En effet, il utilise le backend de la base de données relationnelle pour stocker les données RDF. Cela indique que nous devons adopter la réécriture de SPARQL en SQL (Soussi and Bahaj 2017). Par conséquent, le temps de réponse aux requêtes calculé prend également en compte le temps de réécriture.

5.3. Comparaison des systèmes proposés contre systèmes existants

Afin de comparer les systèmes de stockage RDF proposés avec les systèmes existants, nous ne considérons que les deux meilleures méthodes de chaque groupe : RDFPC et RDFVP représentent le groupe des systèmes proposés et TDB et RDF4JM pour les systèmes existants.

5.3.1. Le temps du chargement

En analysant les résultats de la Figure 22, on remarque que le RDFVP est le meilleur choix lorsque le stockage concerne de petits ensembles de données. En revanche, pour un nombre plus élevé de triples, il est évident que la TDB est plus performante que tous les systèmes testés.

Pour justifier davantage l'efficacité des méthodes proposées par rapport aux méthodes existantes, nous avons effectué une autre analyse expérimentale détaillée en utilisant le jeu de données DBPedia (geo_coordinate) qui contient 1569180 triplets. A cet égard, la Figure 21 montre le temps de chargement en ms pour un nombre croissant de triplets allant de 10% à 100% avec un intervalle de 10% du nombre original de triplets. D'après le résultat, nous pouvons confirmer que le RDFVP donne une bonne performance de temps de chargement sur un petit nombre de triples, tandis que lorsque le nombre de triples augmente, la TDB prouve sa robustesse en termes d'évolutivité et peut être utile pour charger un grand jeu de données.

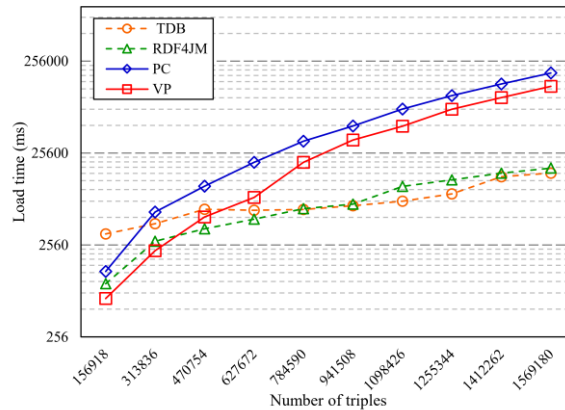


Figure 17: Comparison of the proposed methods against the existing ones in terms of scalability.

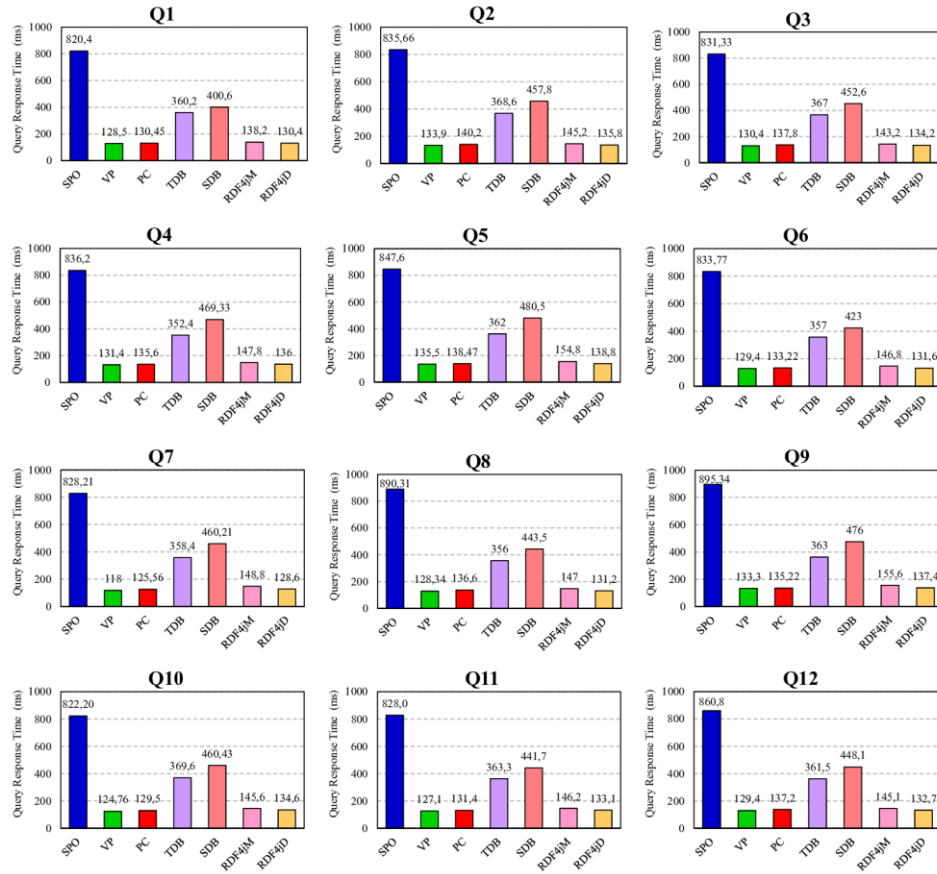


Figure 18: Query response time comparison between the proposed methods and existing ones.

5.3.2. Temps de réponses des requêtes

En comparant les systèmes existants, les résultats obtenus par RDFVP sont plus stables que ceux produits par les autres systèmes existants. En outre, la méthode RDFVP présente le meilleur temps de réponse aux requêtes, ce qui garantit l'utilité de l'implémentation proposée. La raison de ces résultats est liée à deux aspects importants. Le premier est que RDFVP ne supporte pas la stratégie de raisonnement, ce qui signifie que les systèmes qui adoptent l'applicabilité du raisonnement peuvent conduire à une augmentation du temps de réponse. Le deuxième aspect concerne le langage de requête utilisé par le RDFVP, à savoir SQL. En fait, il n'est pas toujours vrai que SQL est plus rapide que SPARQL car cela dépend du système utilisé pour effectuer la requête ainsi que des caractéristiques du jeu de données (Domingue et al. 2011b). Fondamentalement, le RDFVP est capable de fournir une mise en œuvre utile pour l'analyse exploratoire et d'autres applications du web sémantique.

5.4. Discussion

Pour donner la signification statistique des différences de temps de chargement et de temps de réponse aux requêtes des sept méthodes testées, un test non paramétrique de Friedman est effectué. Le test de Friedman (Friedman 1937, 1940) est un test statistique qui utilise le rang de chaque méthode sur chaque échantillon. La statistique de Friedman χ_F^2 est donnée par

$$\chi_F^2 = \frac{12N}{k(k+1)} \left(\sum_{j=1}^k R_j^2 - \frac{k(k+1)^2}{4} \right),$$

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2},$$

où $k = 7$ est le nombre de méthodes testées, N est le nombre d'échantillons, qui est représenté par des ensembles de données dans le cas de la mesure du temps de chargement ($N = 17$) et des requêtes dans le cas du temps de réponse aux requêtes ($N = 12$). R_j est le rang moyen de la méthode j parmi tous les échantillons de test. F_F suit une distribution de Fisher avec $k - 1$ and $(k - 1)(N - 1)$ degrés de liberté.

Pour effectuer le test de Friedman, les tableaux 24 et 25 montrent respectivement les mesures du temps de chargement et du temps de réponse aux requêtes en utilisant sept méthodes. De plus, nous avons reporté sur chaque tableau et pour chaque méthode le rang correspondant, qui est affiché entre parenthèses ($\cdot$).

Data Set	RDFSPO	RDFVP	RDFPC	TDB	SDB	RDF4JM	RDF4JD
allSWGroups	581.1 (5)	280.63 (1)	525.8 (4)	611 (6)	1275.14 (7)	359.9 (2)	404.7 (3)
Wikimovies	545.7 (4)	357.42 (1)	580.6 (5)	650 (6)	1300.34 (7)	463.7 (2)	505.5 (3)
NATS	950 (6)	420.42 (1)	670.6 (4)	866.75 (5)	1999.45 (7)	569.7 (2)	593 (3)
tissues	998.1 (5)	604.31 (1)	895.3 (4)	1084 (6)	2010.23 (7)	648.8 (2)	854.7 (3)
locations	1217.5 (6)	608.16 (1)	898.8 (4)	1093.12 (5)	3500.56 (7)	659.7 (2)	862.5 (3)
VSRR	2535.5 (6)	616.21 (1)	900.3 (4)	1114.5 (5)	4024.45 (7)	664.6 (2)	870.5 (3)
AQMNEHTN	2744.4 (6)	623.14 (1)	950.2 (4)	1133.5 (5)	85000.56 (7)	697.5 (2)	886.9 (3)
CDI	6280.8 (6)	800.35 (1)	1410.8 (4)	1562.37(5)	109958 (7)	807.9 (2)	899 (3)
DHDS	6438.3 (6)	896.42 (2)	1420.4 (5)	1354.37 (4)	115987.7 (7)	866.8 (1)	984.8 (3)
DASH	6841 (6)	1001.63 (2)	1460.8 (4)	1781.87 (5)	132024.4 (7)	914.4 (1)	1300.5 (3)
journals	7214.4 (6)	2576.49 (3)	3790.3 (5)	2055.5 (2)	160546.5 (7)	1467.5 (1)	3215.5 (4)
diseases	13954.7 (6)	4200.14 (3)	6200.5 (5)	2545.5 (2)	180045.5 (7)	2311.2 (1)	4444.5 (4)
enzyme	19885.8 (6)	4410.35 (3)	6700.5 (5)	2668.625 (2)	200560.7 (7)	2500.8 (1)	5158 (4)

geneSymbol	24986.3 (6)	6090.42 (3)	9025.6 (5)	4273.375 (2)	1300022 (7)	3480.4 (1)	8202.8 (4)
protein	32154.7 (6)	7788.2 (3)	15125.8 (5)	4802.75 (2)	1662363 (7)	3878.8 (1)	10162.8 (4)
swdf	38508.8 (6)	9488.78 (3)	17644.3 (5)	5551.875 (1)	2756740 (7)	6423.1 (2)	10576.4 (4)
geo_coordinates_en	300167.7 (6)	119178.7 (3)	190222.2 (5)	15625.25 (1)	26891431 (7)	17549 (2)	123199.8 (4)
Average Rank:	(5.76)	(1.94)	(4.53)	(3.76)	(7.00)	(1.59)	(3.41)

Table 19: Comparison of Load Time (ms) metric of the proposed and existing methods on 17 data sets. The ranks in the parentheses (·) are used in the computation of the Friedman test.

Queries	RDFSPO	RDFVP	RDFPC	TDB	SDB	RDF4JM	RDF4JD
Q1	820.4 (7)	128.5 (1)	130.45 (3)	360.2 (5)	400.6 (6)	138.2 (4)	130.4 (2)
Q2	835.66 (7)	133.9 (1)	140.2 (3)	368.6 (5)	457.8 (6)	145.2 (4)	135.8 (2)
Q3	831.33 (7)	130.4 (1)	137.8 (3)	367 (5)	452.6 (6)	143.2 (4)	134.2 (2)
Q4	836.2 (7)	131.4 (1)	135.6 (2)	352.4 (5)	469.33 (6)	147.8 (4)	136 (3)
Q5	847.6 (7)	135.5 (1)	138.47 (2)	362 (5)	480.5 (6)	154.8 (4)	138.8 (3)
Q6	833.77 (7)	129.4 (1)	133.22 (3)	357 (5)	423 (6)	146.8 (4)	131.6 (2)
Q7	828.21 (7)	118 (1)	125.56 (2)	358.4 (5)	460.21 (6)	148.8 (4)	128.6 (3)
Q8	890.31 (7)	128.34 (1)	136.6 (3)	356 (5)	443.5 (6)	147 (4)	131.2 (2)
Q9	895.34 (7)	133.3 (1)	135.22 (2)	363 (5)	476 (6)	155.6 (4)	137.4 (3)
Q10	822.2 (7)	124.76 (1)	129.5 (2)	369.6 (5)	460.43 (6)	145.6 (4)	134.6 (3)
Q11	860,8 (7)	129,4 (1)	137,2 (3)	361,5 (5)	448,1 (6)	145,1 (4)	132,7 (2)
Q12	828,0 (7)	127,1 (1)	131,4 (2)	363,3 (5)	441,7 (6)	146,2 (4)	133,1 (3)
Average rank:	(7.00)	(1.00)	(2.50)	(5.00)	(6.00)	(4.00)	(2.50)

Table 20 Comparison of Query Response Time (ms) metric of the proposed and existing methods on 12 queries. The ranks in the parentheses (·) are used in the computation of the Friedman test.

Les hypothèses nulles de Friedman considèrent que les sept méthodes sont équivalentes en termes de performances de calcul, notamment le temps de chargement et le temps de réponse aux requêtes. Nous considérons un niveau de confiance $\alpha = 0.05$. Selon le test de Friedman, les résultats du test p-value pour le temps de chargement et le temps de réponse à la requête sont respectivement de 7.62×10^{-16} and 2.91×10^{-13} . Par conséquent, il est évident de rejeter les hypothèses nulles, et les sept méthodes testées sont différentes en termes de performance de calcul pour le temps de chargement et le temps de réponse aux requêtes.

Ensuite, nous utilisons le test post-hoc, à savoir le test de Nemenyi (Demšar 2006; Derrac et al. 2011), pour analyser davantage la performance relative entre les systèmes testés. Les performances de deux méthodes sont significativement différentes si les rangs moyens correspondants diffèrent d'au moins la différence critique (CD) :

$$CD_{\alpha} = q_{\alpha} \sqrt{\frac{k(k+1)}{6N}}.$$

Pour le test de Nemeyi, nous avons la valeur critique tabulée est $q_{\alpha} = 2.949$ à un niveau de signification $\alpha = 0.05$. Par conséquent, les CD pour le temps de chargement et le temps de réponse aux requêtes sont respectivement égaux à 2.1850 ($k=7, N=17$) and 2.6007 ($k=7, N=12$).

Pour illustrer visuellement la performance relative des méthodes proposées (RDFSPO, RDFPC et RDFVP) par rapport aux autres méthodes existantes, les diagrammes CD sur les différentes mesures d'évaluation sont présentés dans la figure 24 (a) et (b), où le rang moyen de chaque méthode de comparaison est signé le long de l'axe (les rangs plus élevés à droite). Dans chaque sous-figure, une ligne épaisse est utilisée pour connecter les méthodes dont les rangs moyens sont inférieurs à la différence critique. Sinon, tous les systèmes qui ne sont pas reliés entre eux sont reconnus pour avoir des performances apparemment différentes.

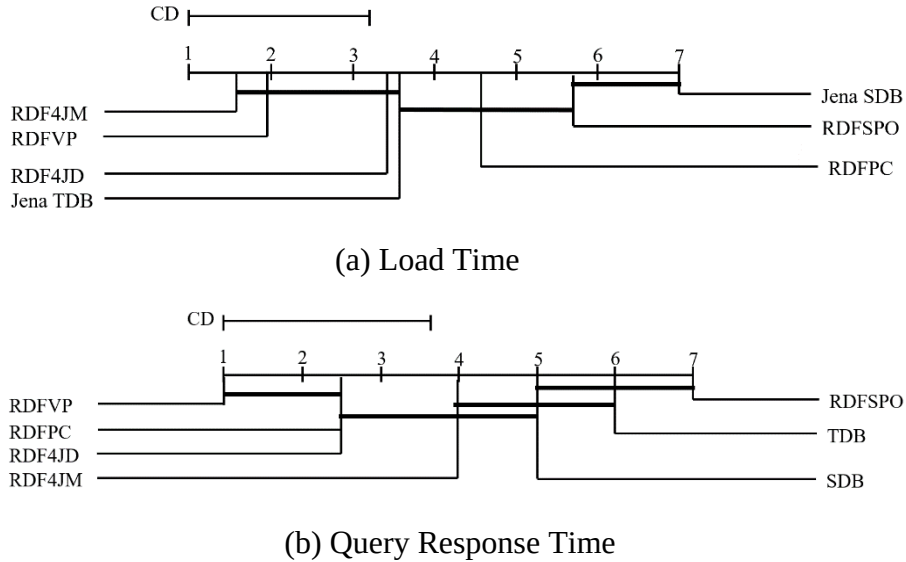


Figure 19: Computational performance comparison of the proposed RDFVP, RDFSPO and RDFPC against the existing methods with Nemeyi Test.

Le test statistique présenté à la figure 10 (a) révèle que le temps de chargement avec RDF4JM est statistiquement meilleur que ceux avec RDFPC, RDFSPO et RDFSDB (as $RDFPC - RDF4JM = 4.52 - 1.58 > 2.18$, $RDFSPO - RDF4JM = 5.76 - 1.58 > 2.18$ and $SDB - RDF4JM = 7.00 - 1.58 > 2.18$). En outre, il convient de noter qu'il n'y a pas de preuves cohérentes pour indiquer des différences statistiques de temps de charge entre le RDFVP proposé et le RDF4JM (as $RDFVP - RDF4JM = 1.94 - 1.58 < 2.18$), la même conclusion vaut pour le VP avec RDF4JD et avec TDB.

Les résultats sont présentés à la Figure 10 (b) et indiquent que le temps de réponse aux requêtes de notre méthode RDFVP est statistiquement meilleur que celui de RDF4JM, SDB, TDB et RDFSPO

(as $\text{RDF4JM}-\text{RDFVP}=4.00-1.00>2.600$, $\text{SDB}-\text{RDFVP}=6.00-1.00>2.600$, $\text{TDB}-\text{RDFVP}=5.00-1.00>2.600$ and $\text{RDFSPO}-\text{RDFVP}=7.00-1.00>2.600$). Cependant, il n'y a pas de preuves cohérentes pour indiquer des différences statistiques de temps de réponse aux requêtes entre RDFVP, RDFPC et RDF4JD (as $\text{RDFPC}-\text{RDFVP}=2.50-1.00 < 2.600$ and $\text{RDF4JD}-\text{RDFVP}=2.50-1.00<2.600$).

En résumé, la méthode RDFVP proposée atteint des performances de calcul très compétitives par rapport aux autres méthodes de pointe en termes de temps de chargement et de temps de réponse aux requêtes.

Conclusion

Enfin, dans ce chapitre, nous avons abordé trois nouvelles implémentations de méthodes non natives pour le stockage des données RDF. Ces implémentations sont respectivement basées sur les approches de table d'énoncés, de table de propriétés et de partitionnement vertical. De plus, nous examinons la question de savoir comment sélectionner des solutions de stockage pratiques et efficaces en fonction des caractéristiques de l'ensemble de données. À cet égard, deux importantes mesures de performance sont fournies, à savoir le temps de chargement et le temps de réponse aux requêtes pour évaluer les systèmes de stockage RDF. Afin de démontrer l'efficacité et la robustesse des systèmes de stockage RDF proposés et existants, les études expérimentales ont été divisées en trois sections. La première a consisté à évaluer les systèmes de stockage RDF proposés. La deuxième section a été consacrée à l'évaluation des systèmes de stockage RDF existants. Dans la troisième section, nous fournissons une comparaison entre les systèmes de stockage RDF proposés et existants. En outre, pour donner une raison significative aux résultats obtenus, nous avons appliqué une analyse statistique basée sur le test de Friedman, suivi d'un test post hoc de Nemenyi, afin d'explorer plus avant les performances statistiques des systèmes. Dans les travaux futurs, nous souhaitons adopter des algorithmes d'apprentissage automatique pour prédire le système le plus approprié pour un ensemble de données spécifique. Plus précisément, nous prévoyons de contourner les approches traditionnelles d'estimation du temps de chargement et du temps de réponse aux requêtes, qui sont basées sur des statistiques relatives au jeu de données RDF sous-jacent. Dans ce contexte, nous montrerons comment modéliser ces deux mesures en tant que vecteurs de caractéristiques afin de les utiliser avec précision comme entrée pour des algorithmes d'apprentissage automatique.

Chapter4. Conclusions et Perspectives

1. Conclusions

L'objectif principal de cette thèse était de proposer un cycle de vie complet pour générer des ontologies à partir d'une base de données relationnelle. À cet égard, nous avons commencé cette thèse en fournissant quelques informations de base sur le graphe de connaissances. En fait, le graphe de connaissances (KG) représente une collection de descriptions reliées entre elles d'entités - objets et événements du monde réel, ou concepts abstraits (par exemple, des documents) où ces entités ont des définitions basées sur une sémantique formelle. L'avantage d'utiliser des définitions formelles est qu'elles permettent aux personnes et aux ordinateurs de traiter les entités de manière efficace et sans ambiguïté. La seule façon de fournir des définitions formelles aux entités est d'utiliser des ontologies. Par conséquent, les ontologies représentent les éléments fondamentaux du graphe de connaissances et peuvent être considérées comme le schéma du graphe. Elles assurent une compréhension partagée des données et de leurs significations.

D'une manière générale, le graphe de connaissances est basé sur trois couches principales qui sont : l'acquisition de connaissances, le stockage de connaissances et la consommation de connaissances. Plus profondément, le développement des ontologies réside dans la couche d'acquisition des connaissances. En effet, il existe deux approches principales pour construire des ontologies, soit à partir de zéro, soit en utilisant des méthodes d'apprentissage d'ontologies. Cette dernière approche permet de construire des ontologies à partir de sources non ontologiques telles que des bases de données relationnelles, des dictionnaires, des documents textuels, des pages web, etc. Dans cette thèse, nous nous sommes concentrés sur les bases de données relationnelles comme source d'information pour l'apprentissage d'ontologies. Dans cette thèse, nous nous sommes concentrés sur les bases de données relationnelles comme source d'information pour l'apprentissage d'une ontologie : Modèle conceptuel, modèle logique et modèle physique.

Malgré les progrès significatifs réalisés au cours des dernières années et le grand nombre d'approches proposées, il reste encore de nombreuses questions qui n'ont pas été suffisamment traitées. Dans ce contexte, cette thèse a apporté des réponses aux problématiques spécifiques suivantes :

- Toutes les études proposées n'ont pas défini un cycle de vie pour décrire les scénarios les plus courants qui se produisent au cours du processus de génération d'ontologies à partir de la BDR.

- Tous les travaux existants se concentrent uniquement sur la construction de A-Box ou T-Box et ignorent le processus d'intégration entre ces deux composants.
- La majorité des études ne couvrent pas et ne comprennent pas la sémantique maximale résidant dans la base de données, comme la contrainte par défaut et la transitivité...
- Un autre défi qui s'est présenté après avoir terminé le processus de développement de l'ontologie est l'évaluation. La majorité des travaux ignorent ce sujet parce que l'évaluation de l'ontologie est une tâche très difficile qui est basée sur la subjectivité. En fait, l'ontologie résultante doit être évaluée afin de déterminer sa qualité avant son utilisation ou sa réutilisation.
- Enfin, une réponse à la question suivante est abordée : comment choisir une stratégie appropriée pour le stockage des ontologies.

Dans cette thèse, nous avons proposé des réponses à ces problématiques à travers trois contributions

Contribution 1: Proposing a new life cycle

Nous avons proposé un nouveau cycle de vie pour l'apprentissage d'ontologies à partir de RDBs basé sur les exigences de l'ingénierie logicielle et des projets big data. De manière générale, ce cycle de vie comprend quatre phases principales qui peuvent être soit en avant soit en arrière dans un mouvement : Découverte, Préparation, Développement, et Évaluation. Nous avons ensuite décrit une nouvelle méthode pour construire une ontologie à partir d'une base de données relationnelle qui peut être considérée comme l'une des mises en œuvre du cycle de vie prédéfini. Enfin, nous avons suggéré trois nouvelles sémantiques qui peuvent être extraites de RDB : Contrainte par défaut, transitivité et héritage.

Contribution2: Ontology Evaluation

Nous avons présenté un processus d'évaluation basé sur deux catégories : La vérification et la validation. La première consiste à vérifier la qualité de l'ontologie résultante (l'ontologie construite à partir de la BDR). Comme nous le savons, l'ontologie est représentée par le schéma (T-Box) et les instances (A-Box). C'est la raison pour laquelle la vérification est divisée en deux phases : la vérification de la boîte A et la vérification de la boîte T. D'autre part, la validation vise à aider l'ontologue à mieux comprendre la complexité des ontologies existantes, dans un domaine spécifique, afin de justifier la réutilisation ou le maintien de ces ontologies.

A notre connaissance, il s'agit de la première étude qui couvre le processus d'évaluation en tenant compte de toutes ces questions.

Contribution 3: Ontology Storage

Nous avons proposé trois nouvelles implémentations de méthodes non natives. Ces implémentations sont respectivement basées sur les approches de table d'énoncés, de table de propriétés et de partitionnement vertical. Nous avons présenté une étude empirique pour aborder les questions de sélection de la stratégie la plus pertinente pour le stockage des ontologies. À cet égard, deux importantes mesures de performance sont fournies, qui comprennent le temps de chargement et le temps de réponse aux requêtes pour évaluer les systèmes de stockage RDF. En outre, pour apporter une raison significative aux résultats obtenus, nous avons appliqué une analyse statistique basée sur le test de Friedman suivi d'un test post hoc de Nemenyi afin d'explorer plus avant le système le plus performant statistiquement.

Enfin, nous pouvons confirmer que cette recherche a atteint les objectifs fixés dans l'introduction générale en abordant les questions critiques dans le processus de développement de l'ontologie.

2. Orientations futures de la recherche

Le travail présenté dans cette thèse a démontré le profond potentiel d'amélioration de l'efficacité et de l'efficacité du processus de développement d'ontologies à partir de bases de données relationnelles. Cependant, il y a encore beaucoup d'efforts à faire pour améliorer ce processus. Par conséquent, afin d'approfondir cette recherche, les travaux futurs suivants sont recommandés :

- Affiner le cycle de vie : Dans cette thèse, nous avons présenté un nouveau cycle de vie pour construire des ontologies à partir de bases de données relationnelles. Cependant, nous devons améliorer ce cycle de vie pour répondre aux problèmes suivants : Le nettoyage des données, l'alignement des ontologies, l'intégration des ontologies, etc.
- Un autre défi sur lequel nous devons travailler est le processus d'évaluation. En fait, l'évaluation d'une ontologie est un processus subjectif et il n'existe pas de métrique standard qui puisse être appliquée pour mesurer la qualité de l'ontologie construite. À cet égard, nous nous concentrerons sur la construction d'un cadre pour l'évaluation des ontologies en fonction des exigences de l'utilisateur final.
- La troisième perspective de ce travail est de combiner l'apprentissage automatique avec les ontologies afin de répondre à la question suivante : Comment pouvons-nous renforcer les modèles d'apprentissage automatique en utilisant des ontologies ?

List of Publications

- Ben Mahria, B., Chaker, I. & Zahi, A. An empirical study on the evaluation of the RDF storage systems. *J Big Data* **8**, 100 (2021). <https://doi.org/10.1186/s40537-021-00486-y>
- Ben Mahria, B., Chaker, I. & Zahi, A. A novel approach for learning ontology from relational database: from the construction to the evaluation. *J Big Data* **8**, 25 (2021). <https://doi.org/10.1186/s40537-021-00412-2>
- Ben Mahria, B.; Chaker, I. and Zahi, A. (2020). Measuring Design Complexity of Cultural Heritage Ontologies. In *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - KEOD*, ISBN 978-989-758-474-9; ISSN 2184-3228, pages 133-140. DOI: 10.5220/0010016501330140
- Benmahria B., Chaker I., Zahi A. (2019) Validation and Evaluation of the Mapping Process for Generating Ontologies from Relational Databases. In: Rocha Á., Adeli H., Reis L., Costanzo S. (eds) *New Knowledge in Information Systems and Technologies. WorldCIST'19 2019. Advances in Intelligent Systems and Computing*, vol 930. Springer, Cham. https://doi.org/10.1007/978-3-030-16181-1_32

References

- Aasman J (2006) Allegro graph: RDF triple database. Cidade: Oakland Franz Incorporated 17:
- Abadi DJ, Marcus A, Madden SR, Hollenbach K (2007a) Scalable semantic web data management using vertical partitioning. In: Proceedings of the 33rd international conference on Very large data bases. VLDB Endowment, pp 411–422
- Abadi DJ, Marcus A, Madden SR, Hollenbach K (2007b) Scalable semantic web data management using vertical partitioning. In: Proceedings of the 33rd international conference on Very large data bases. pp 411–422
- Abbasi R, Staab S, Cimiano P (2007) Organizing resources on tagging systems using t-org. In: 4th European Semantic Web Conference. pp 97–110
- Al-Ajlan A (2015) The comparison between forward and backward chaining. International Journal of Machine Learning and Computing 5:106
- Alaoui K (2019) A categorization of rdf triplestores. In: Proceedings of the 4th International Conference on Smart City Applications. pp 1–7
- Al-Arfaj A, Al-Salman A (2015) Ontology construction from text: challenges and trends. International Journal of Artificial Intelligence and Expert Systems (IJAE) 6:15–26
- Alexaki S, Christophides V, Karvounarakis G, et al (2001) The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. In: SemWeb
- Alexopoulos P (2020) Semantic Modeling for Data. O'Reilly Media
- An Y, Mylopoulos J (2005) Translating XML web data into ontologies. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". Springer, pp 967–976
- Angles R (2018) The Property Graph Database Model. In: AMW
- Antoniou G, Van Harmelen F (2004) A semantic web primer. MIT press
- Astrova I (2004) Reverse engineering of relational databases to ontologies. In: European Semantic Web Symposium. Springer, pp 327–341
- Astrova I, Korda N, Kalja A (2007) Rule-based transformation of SQL relational databases to OWL ontologies. In: Proceedings of the 2nd International Conference on Metadata & Semantics Research. Citeseer, pp 415–424
- Atre M, Srinivasan J, Hendler JA (2009) BitMat: A main memory RDF triple store. Tetherless World Constellation, Rensselaer Polytechnic Institute, Troy NY

- Auer S, Dietzold S, Lehmann J, et al (2009) Triplify: light-weight linked data publication from relational databases. In: Proceedings of the 18th international conference on World wide web. pp 621–630
- Baker T (2012) Libraries, languages of description, and linked data: a Dublin Core perspective. Library Hi Tech
- Bakkas J, Bahaj M, Marzouk A (2013) Direct migration method of rdb to ontology while keeping semantics. International Journal of Computer Applications 65:
- Barrasa J, Corcho Ó, Gómez-Pérez A (2004) R2O, an extensible and semantically based database-to-ontology mapping language. In: Proceedings of the 2nd Workshop on Semantic Web and Databases, Toronto, Canada
- Barrasa J, Corcho O, Gómez-Pérez A (2003) Fund Finder: A case study of database-to-ontology mapping. In: Semantic Integration Workshop (SI-2003). p 9
- Beckett D (2002) The design and implementation of the Redland RDF application framework. Computer Networks 39:577–588
- Bellomarini L, Fakhoury D, Gottlob G, Sallinger E (2019) Knowledge graphs and enterprise AI: the promise of an enabling technology. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE). IEEE, pp 26–37
- Bizer C (2003) D2r map-a database to rdf mapping language
- Bizer C, Cyganiak R (2006) D2r server-publishing relational databases on the semantic web. In: Poster at the 5th international semantic web conference
- Bizer C, Seaborne A (2004) D2RQ-treating non-RDF databases as virtual RDF graphs. In: Proceedings of the 3rd international semantic web conference (ISWC2004). Proceedings of ISWC2004
- Blanco R, Cambazoglu BB, Mika P, Torzec N (2013) Entity recommendations in web search. In: International Semantic Web Conference. Springer, pp 33–48
- Blumauer A, Nagy H, Nagy H (2020a) The Knowledge Graph Cookbook. edition mono/monochrom
- Blumauer A, Nagy H, Nagy H (2020b) The Knowledge Graph Cookbook. edition mono/monochrom
- Bonatti PA, Decker S, Polleres A, Presutti V (2019) Knowledge graphs: New directions for knowledge representation on the semantic web (dagstuhl seminar 18371). In: Dagstuhl Reports. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik

- Brank J, Grobelnik M, Mladenic D (2005) A survey of ontology evaluation techniques. In: Proceedings of the conference on data mining and data warehouses (SiKDD 2005). Citeseer Ljubljana, Slovenia, pp 166–170
- Brewster C, Alani H, Dasmahapatra S, Wilks Y (2004) Data driven ontology evaluation
- Broekstra J, Kampman A, Van Harmelen F (2002) Sesame: A generic architecture for storing and querying rdf and rdf schema. In: International semantic web conference. Springer, pp 54–68
- Buccella A, Penabad MR, Rodriguez FJ, et al (2004) From relational databases to OWL ontologies. In: Proceedings of the 6th National Russian Research Conference
- Calero C, Ruiz F, Piattini M (2006) Ontologies for software engineering and software technology. Springer Science & Business Media
- Candela G, Escobar P, Carrasco RC, Marco-Such M (2020) Evaluating the quality of linked open data in digital libraries. Journal of Information Science 0165551520930951
- Carlson A, Betteridge J, Wang RC, et al (2010) Coupled semi-supervised learning for information extraction. In: Proceedings of the third ACM international conference on Web search and data mining. pp 101–110
- Chen JX, Reformat MZ (2014) Learning categories from linked open data. In: International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems. Springer, pp 396–405
- Chiang RH, Barron TM, Storey VC (1994) Reverse engineering of relational databases: Extraction of an EER model from a relational database. Data & knowledge engineering 12:107–142
- Chujai P, Kerdprasop N, Kerdprasop K (2014a) On transforming the ER model to ontology using protégé OWL tool. International Journal of Computer Theory and Engineering 6:484
- Chujai P, Kerdprasop N, Kerdprasop K (2014b) On transforming the ER model to ontology using protégé OWL tool. International Journal of Computer Theory and Engineering 6:484
- Copeland GP, Khoshafian SN (1985) A decomposition storage model. In: Acm Sigmod Record. ACM, pp 268–279
- Coronel C, Morris S (2016) Database systems: design, implementation, & management. Cengage Learning
- Corwin J, Silberschatz A, Miller PL, Marenco L (2007) Dynamic tables: an architecture for managing evolving, heterogeneous biomedical data in relational database management systems. Journal of the American Medical Informatics Association 14:86–93
- Curé O, Blin G (2014) RDF database systems: triples storage and SPARQL query processing. Morgan Kaufmann

- Dadjoo M, Kheirkhah E (2015) An approach for transforming of relational databases to OWL ontology. arXiv preprint arXiv:150205844
- Davoudian A, Chen L, Liu M (2018) A survey on NoSQL stores. *ACM Computing Surveys (CSUR)* 51:1–43
- de Laborda CP, Conrad S (2005) Relational. OWL: a data and schema representation format based on OWL. In: *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling-Volume 43*. Citeseer, pp 89–96
- de Medeiros LF, Priyatna F, Corcho O (2015a) MIRROR: Automatic R2RML mapping generation from relational databases. In: *International Conference on Web Engineering*. Springer, pp 326–343
- de Medeiros LF, Priyatna F, Corcho O (2015b) MIRROR: Automatic R2RML mapping generation from relational databases. In: *International Conference on Web Engineering*. Springer, pp 326–343
- Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research* 7:1–30
- Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation* 1:3–18
- Dessi D, Osborne F, Recupero DR, et al (2021) Generating knowledge graphs by employing Natural Language Processing and Machine Learning techniques within the scholarly domain. *Future Generation Computer Systems* 116:253–264
- Dimou A, Vander Sande M, Colpaert P, et al (2014) RML: a generic language for integrated RDF mappings of heterogeneous data. In: *Ldow*
- Din AI (2014) *Structured query language (SQL) A practical Introduction*
- Domingue J, Fensel D, Hendler JA (2011a) *Handbook of semantic web technologies*. Springer Science & Business Media
- Domingue J, Fensel D, Hendler JA (2011b) *Handbook of semantic web technologies*. Springer Science & Business Media
- DuCharme B (2013) *Learning SPARQL: querying and updating with SPARQL 1.1*. O'Reilly Media, Inc.
- Ehrlinger L, Wöß W (2016) Towards a Definition of Knowledge Graphs. *SEMANTiCS (Posters, Demos, SuCCESS)* 48:1–4

- El Alami A, Bahaj M The Migration of a Conceptual Object Model COM (Conceptual Data Model CDM, Unified Modeling Language UML class diagram...) to the Object Relational Database ORDB. MAGNT Research Report (ISSN 1444-8939) 2:318–32
- Erling O, Mikhailov I (2010) Virtuoso: RDF support in a native RDBMS. In: Semantic Web Information Management. Springer, pp 501–519
- Erling O, Mikhailov I (2009) RDF Support in the Virtuoso DBMS. In: Networked Knowledge-Networked Media. Springer, pp 7–24
- Fahad M (2008a) Er2owl: Generating owl ontology from er diagram. In: International Conference on Intelligent Information Processing. Springer, pp 28–37
- Fahad M (2008b) Er2owl: Generating owl ontology from er diagram. In: International Conference on Intelligent Information Processing. Springer, pp 28–37
- Färber M, Bartscherer F, Menne C, Rettinger A (2018) Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. Semantic Web 9:77–129
- Faye D, Curé O, Blin G, Thiam C (2012) RDF triples management in roStore. In: IC 2011, 22èmes Journées francophones d'Ingénierie des Connaissances. pp 755–770
- Fernández M, Overbeeke C, Sabou M, Motta E (2009) What makes a good ontology? A case-study in fine-grained knowledge reuse. In: Asian Semantic Web Conference. Springer, pp 61–75
- Fernández-López M, Gómez-Pérez A, Juristo N (1997) Methontology: from ontological art towards ontological engineering
- Fletcher GH, Beck PW (2009) Scalable indexing of RDF graphs for efficient join processing. In: Proceedings of the 18th ACM conference on Information and knowledge management. ACM, pp 1513–1516
- Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the american statistical association 32:675–701
- Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. The Annals of Mathematical Statistics 11:86–92
- Gangemi A, Guarino N, Masolo C, Oltramari A (2003a) Sweetening wordnet with dolce. AI magazine 24:13–13
- Gangemi A, Navigli R, Velardi P (2003b) The OntoWordNet Project: extension and axiomatization of conceptual relations in WordNet. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". Springer, pp 820–838
- Garcia R, Celma O (2005) Semantic Integration and Retrieval of Multimedia Metadata. In: SemAnnot@ ISWC

- Gayo JEL, Prud'Hommeaux E, Boneva I, Kontokostas D (2017a) Validating RDF data. *Synthesis Lectures on Semantic Web: Theory and Technology* 7:1–328
- Gayo JEL, Prud'Hommeaux E, Boneva I, Kontokostas D (2017b) Validating RDF data. *Synthesis Lectures on Semantic Web: Theory and Technology* 7:1–328
- Ghawi R, Cullot N (2007a) Database-to-ontology mapping generation for semantic interoperability. In: *Third International Workshop on Database Interoperability (InterDB 2007)*
- Ghawi R, Cullot N (2007b) Database-to-ontology mapping generation for semantic interoperability. In: *VDBL'07 conference, VLDB Endowment ACM*. pp 1–8
- Golbeck J, Rothstein M (2008) Linking Social Networks on the Web with FOAF: A Semantic Web Case Study. In: *AAAI*. pp 1138–1143
- Gruber TR (1993) A translation approach to portable ontology specifications. *Knowledge acquisition* 5:199–220
- Grüninger M, Fox MS (1995) The role of competency questions in enterprise engineering. In: *Benchmarking—Theory and practice*. Springer, pp 22–31
- Guha RV (2004) Rdfdb: An rdf database. 2000. Disponível: <http://rdfdb.sourceforge.net/>[capturado em 15 nov 2003]
- Guha RV, Brickley D, Macbeth S (2016) Schema. org: evolution of structured data on the web. *Communications of the ACM* 59:44–51
- Gutierrez C, Hurtado CA, Mendelzon AO, Pérez J (2011) Foundations of semantic web databases. *Journal of Computer and System Sciences* 77:520–541
- Hahn U (2003) Turning informal thesauri into formal ontologies: a feasibility study on biomedical knowledge re-use. *Comparative and functional genomics* 4:94–97
- Hakkarainen S, Hella L, Strasunskas D, Tuxen S (2006) A semantic transformation approach for ISO 15926. In: *International Conference on Conceptual Modeling*. Springer, pp 281–290
- Harris S, Gibbins N (2003) 3store: Efficient bulk RDF storage
- Harris S, Lamb N, Shadbolt N (2009a) 4store: The design and implementation of a clustered RDF store. In: *5th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009)*. pp 94–109
- Harris S, Lamb N, Shadbolt N (2009b) 4store: The design and implementation of a clustered RDF store. In: *5th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009)*. pp 94–109

- Harth A, Decker S (2005) Optimized index structures for querying rdf from the web. In: Third Latin American Web Congress (LA-WEB'2005). IEEE, pp 10–pp
- Harth A, Umbrich J, Hogan A, Decker S (2007) Yars2: A federated repository for querying graph structured data from the web. In: The Semantic Web. Springer, pp 211–224
- Hassan M, Bansal SK (2018) RDF Data Storage Techniques for Efficient SPARQL Query Processing Using Distributed Computation Engines. In: 2018 IEEE International Conference on Information Reuse and Integration (IRI). IEEE, pp 323–330
- He-ping C, Lu H, Bin C (2008) Research and implementation of ontology automatic construction based on relational database. In: 2008 International Conference on Computer Science and Software Engineering. IEEE, pp 1078–1081
- Hepp M (2006) Products and services ontologies: a methodology for deriving OWL ontologies from industrial categorization standards. International Journal on Semantic Web and Information Systems (IJSWIS) 2:72–99
- Hepp M (2007) Possible ontologies: How reality constrains the development of relevant ontologies. IEEE Internet Computing 11:90–96
- Hepp M (2008a) Goodrelations: An ontology for describing products and services offers on the web. In: International conference on knowledge engineering and knowledge management. Springer, pp 329–346
- Hepp M (2008b) Goodrelations: An ontology for describing products and services offers on the web. In: International conference on knowledge engineering and knowledge management. Springer, pp 329–346
- Hepp M, De Bruijn J (2007) GenTax: A generic methodology for deriving OWL and RDF-S ontologies from hierarchical classifications, thesauri, and inconsistent taxonomies. In: European semantic web conference. Springer, pp 129–144
- Hert M, Reif G, Gall HC (2011) A comparison of RDB-to-RDF mapping languages. In: Proceedings of the 7th International Conference on Semantic Systems. pp 25–32
- Hlomani H, Stacey D (2014) Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: A survey. Semantic Web Journal 1:1–11
- Hogan A, Blomqvist E, Cochez M, et al (2020) Knowledge graphs. arXiv preprint arXiv:200302320
- Hyvönen E, Viljanen K, Tuominen J, Seppälä K (2008) Building a national semantic web ontology and ontology service infrastructure—the FinnONTO approach. In: European Semantic Web Conference. Springer, pp 95–109

- Janik M, Kochut K (2005) BRAHMS: a workbench RDF store and high performance memory system for semantic association discovery. In: International Semantic Web Conference. Springer, pp 431–445
- Jimborean I, Groza A (2014) Ranking ontologies in the ontology building competition boc 2014. In: 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP). IEEE, pp 75–82
- Jones M, Alani H (2006) Content-based ontology ranking
- Jun H-G, Im D-H (2020) Semantics-Pserving RDB2RDF Data Transformation Using Hierarchical Direct Mapping. *Applied Sciences* 10:7070
- Karvinen P, Díaz-Rodríguez N, Grönroos S, Lilius J (2019) RDF Stores for Enhanced Living Environments: An Overview. In: Enhanced Living Environments. Springer, pp 19–52
- Khan H, Ali M, Ngomo A-CN, Saleem M (2020) When is the Peak Performance Reached? An Analysis of RDF Triplestores
- Khan ZC, Mashiane T (2014) An analysis of Facebook’s graph search. In: 2014 Information Security for South Africa. IEEE, pp 1–8
- Kiryakov A, Bishoa B, Ognyanoff D, et al (2010) The features of BigOWLIM that enabled the BBC’s World Cup website. In: Workshop on Semantic Data Management (SemData). pp 13–17
- Kolas D, Emmons I, Dean M (2009) Efficient linked-list rdf indexing in parliament. *SSWS* 9:17–32
- Konstantinou N, Spanos D-E (2015) Materializing the web of linked data. Springer
- Kuznetsov SO, Poelmans J (2013) Knowledge representation and processing with formal concept analysis. *Wiley interdisciplinary reviews: Data mining and knowledge discovery* 3:200–215
- Lauser B, Sini M, Liang A, et al (2006) From AGROVOC to the Agricultural Ontology Service/Concept Server. An OWL model for creating ontologies in the agricultural domain. In: Dublin Core Conference Proceedings. Dublin Core DCMI
- Li M, Du X-Y, Wang S (2005) Learning ontology from relational database. In: 2005 International Conference on Machine Learning and Cybernetics. IEEE, pp 3410–3415
- Lin L, Xu Z, Ding Y (2013) OWL Ontology Extraction from Relational Databases via Database Reverse Engineering. *JSW* 8:2749–2760
- Lourdusamy R, John A (2018a) A review on metrics for ontology evaluation. In: 2018 2nd International Conference on Inventive Systems and Control (ICISC). IEEE, pp 1415–1421

- Lourdusamy R, John A (2018b) A review on metrics for ontology evaluation. In: 2018 2nd International Conference on Inventive Systems and Control (ICISC). IEEE, pp 1415–1421
- Lourdusamy R, Mattam XJ (2021) Resource description framework based semantic knowledge graph for clinical decision support systems. In: Web Semantics. Elsevier, pp 69–86
- Lubyte L, Tessaris S (2007a) Extracting Ontologies from Relational Databases. In: Description Logics
- Lubyte L, Tessaris S (2007b) Extracting Ontologies from Relational Databases. In: Description Logics
- Ma Z, Capretz MA, Yan L (2016) Storing massive Resource Description Framework (RDF) data: a survey. *The Knowledge Engineering Review* 31:391–413
- Maala MZ, Delteil A, Azough A (2007) A Conversion Process From Flickr Tags to RDF Descriptions. In: SAW. Citeseer
- Maedche A, Staab S (2004a) Ontology learning. In: Handbook on ontologies. Springer, pp 173–190
- Maedche A, Staab S (2004b) Ontology learning. In: Handbook on ontologies. Springer, pp 173–190
- Maedche A, Staab S (2002) Measuring similarity between ontologies. In: International Conference on Knowledge Engineering and Knowledge Management. Springer, pp 251–263
- Mahdisoltani F, Biega J, Suchanek F (2014) Yago3: A knowledge base from multilingual wikipe-dias. In: 7th biennial conference on innovative data systems research. CIDR Conference
- MahmoudiNasab H, Sakr S (2012) AdaptRDF: adaptive storage management for RDF databases. *International Journal of Web Information Systems* 8:234–250
- McBride B, Reynolds D (2020) Survey of time series database technology
- McGlothlin J, Khan L (2009a) RDFJoin: A scalable data model for persistence and efficient querying of RDF datasets. *Database*
- McGlothlin JP, Khan LR (2009b) RDFKB: efficient support for RDF inference queries and knowledge management. In: Proceedings of the 2009 International Database Engineering & Applications Symposium. ACM, pp 259–266
- McGuinness DL, Van Harmelen F (2004) OWL web ontology language overview. W3C recommendation 10:2004
- Médini L (2020) Semantic Web reasoning

- Michel F, Montagnat J, Zucker CF (2014) A survey of RDB to RDF translation approaches and tools. PhD Thesis, I3S
- Modoni GE, Sacco M, Terkaj W (2014) A survey of RDF store solutions. In: 2014 International Conference on Engineering, Technology and Innovation (ICE). IEEE, pp 1–7
- Myroshnichenko I, Murphy MC (2009a) Mapping ER schemas to OWL ontologies. In: 2009 IEEE International Conference on Semantic Computing. IEEE, pp 324–329
- Myroshnichenko I, Murphy MC (2009b) Mapping ER schemas to OWL ontologies. In: Semantic Computing, 2009. ICSC'09. IEEE International Conference on. IEEE, pp 324–329
- Neto LET, Vidal VMP, Casanova MA, Monteiro JM (2013) R2RML by assertion: A semi-automatic tool for generating customised R2RML mappings. In: Extended Semantic Web Conference. Springer, pp 248–252
- Noy N, Gao Y, Jain A, et al (2019) Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it's done. Queue 17:48–75
- Noy NF, McGuinness DL (2001) Ontology development 101: A guide to creating your first ontology. Stanford knowledge systems laboratory technical report KSL-01-05 and ...
- Ntavelis E, Romero A, Kastanis I, et al (2020) Sesame: Semantic editing of scenes by adding, manipulating or erasing objects. In: European Conference on Computer Vision. Springer, pp 394–411
- Nyulas C, O'connor M, Tu S (2007) DataMaster—a plug-in for importing schemas and data from relational databases into Protege. In: 10th international Protégé conference. Citeseer, pp 15–18
- Obrenović N, Luković I (2014) An approach to consolidation of database check constraints. ICIST 2014
- Ognyanoff RV, Momtchev V, Balev B, Peikov I (2009) D5. 5.2 Validation goals and metrics for the LarKC platform
- ORDYSISKI T (2011) Ontology of E-Commerce Solution. Studia i Materiały Polskiego Stowarzyszenia Zarządzania Wiedza/Studies & Proceedings Polish Association for Knowledge Management 384–395
- Orme AM, Tao H, Etzkorn LH (2006) Coupling metrics for ontology-based system. IEEE software 23:102–108
- Ouyang L, Zou B, Qu M, Zhang C (2011) A method of ontology evaluation based on coverage, cohesion and coupling. In: 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). IEEE, pp 2451–2455

- Pak J, Zhou L (2009) A framework for ontology evaluation. In: Workshop on E-Business. Springer, pp 10–18
- Pan JZ, Vetere G, Gomez-Perez JM, Wu H (2017a) Exploiting linked data and knowledge graphs in large organisations. Springer
- Pan JZ, Vetere G, Gomez-Perez JM, Wu H (2017b) Exploiting linked data and knowledge graphs in large organisations. Springer
- Pan Z, Zhu T, Liu H, Ning H (2018) A survey of RDF management technologies and benchmark datasets. *Journal of Ambient Intelligence and Humanized Computing* 9:1693–1704
- Paulheim H (2017) Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web* 8:489–508
- Perez-Martin J, Bustos B, Saldana M (2020) Semantic Search of Memes on Twitter. arXiv preprint arXiv:200201462
- Po L, Bikakis N, Desimoni F, Papastefanatos G (2020) Linked Data Visualization: Techniques, Tools, and Big Data. *Synthesis Lectures on Semantic Web: Theory and Technology* 10:1–157
- Press R (2008) Ontology and database mapping: a survey of current implementations and future directions. *Journal of Web Engineering* 7:001–024
- Rawat DS, Kashyap NK (2017) Graph database: a complete GDBMS survey. *Int J* 3:217–226
- Reggiori A, van Gulik D-W, Bjelogrljic Z (2003) Indexing and retrieving Semantic Web resources: the RDFStore model. In: SWAD-Europe Workshop on Semantic Web Storage and Retrieval. Citeseer, pp 13–14
- Rodriguez-Muro M, Rezk M (2015) Efficient SPARQL-to-SQL with R2RML mappings. *Journal of Web Semantics* 33:141–169
- Russo G, Anastasio F, Pipitone A, et al (2012) VEBO: Validation of ER diagrams through ontologies and WordNet. In: 2012 IEEE Sixth International Conference on Semantic Computing. IEEE, pp 342–344
- Salazar RC, Liagre F, Mougenot I, et al (2021) A Software Application with Ontology-Based Reasoning for Agroforestry. *Metadata and Semantic Research* 1355:282
- Salvadores M, Correndo G, Omitola T, et al (2010) 4s-reasoner: Rdfs backward chained reasoning support in 4store. In: 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. IEEE, pp 261–264
- Schmidt M, Hornung T, Lausen G, Pinkel C (2009) SP²Bench: a SPARQL performance benchmark. In: 2009 IEEE 25th International Conference on Data Engineering. IEEE, pp 222–233

SCHULZ UHahn TOWARDS A BROAD-COVERAGE BIOMEDICAL ONTOLOGY BASED ON DESCRIPTION LOGICS

- Sedighi SM, Javidan R (2012a) A novel method for improving the efficiency of automatic construction of ontology from a relational database. *International Journal of Physical Sciences* 7:2085–2092
- Sedighi SM, Javidan R (2012b) A novel method for improving the efficiency of automatic construction of ontology from a relational database. *Int J Phys Sci* 7:2085–2092
- Sequeda JF, Tirmizi SH, Corcho O, Miranker DP (2011) Survey of directly mapping sql databases to the semantic web. *The Knowledge Engineering Review* 26:445–486
- Services EE (2015) *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*. Wiley
- Sicilia M-Á, Rodríguez D, García-Barriocanal E, Sánchez-Alonso S (2012) Empirical findings on ontology metrics. *Expert Systems with Applications* 39:6706–6711
- Singh G, Upadhyay D, Atre M (2018) Efficient RDF dictionaries with B+ trees. In: *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*. ACM, pp 128–136
- Soergel D, Lauser B, Liang A, et al (2004) Reengineering thesauri for new applications: the AGROVOC example. *Journal of digital information* 4:1–23
- Soussi N, Bahaj M (2017) Semantics preserving SQL-to-SPARQL query translation for Nested Right and Left Outer Join. *Journal of applied research and technology* 15:504–512
- Spanos D-E, Stavrou P, Mitrou N (2012) Bringing relational databases into the semantic web: A survey. *Semantic Web* 3:169–209
- Sporny M, Longley D, Kellogg G, et al (2014) JSON-LD 1.0. W3C recommendation 16:41
- Sure Y, Staab S, Studer R (2004) On-to-knowledge methodology (OTKM). In: *Handbook on ontologies*. Springer, pp 117–132
- Swartout B, Patil R, Knight K, Russ T (1996) Toward distributed use of large-scale ontologies. In: *Proc. of the Tenth Workshop on Knowledge Acquisition for Knowledge-Based Systems*. pp 138–148
- Swartz A (2002) Musicbrainz: A semantic web service. *IEEE Intelligent Systems* 17:76–77
- Tartir S, Arpinar IB, Moore M, et al (2005) OntoQA: Metric-based ontology quality analysis
- Telnarova Z (2010) Relational database as a source of ontology creation. In: *Proceedings of the International Multiconference on Computer Science and Information Technology*. IEEE, pp 135–139

- ter Horst HJ (2005) Combining RDF and part of OWL with rules: Semantics, decidability, complexity. In: International Semantic Web Conference. Springer, pp 668–684
- Thakker D, Osman T, Gohil S, Lakin P (2010) A pragmatic approach to semantic repositories benchmarking. In: Extended Semantic Web Conference. Springer, pp 379–393
- Thalheim B (2013) Entity-relationship modeling: foundations of database technology. Springer Science & Business Media
- Tirmizi SH, Sequeda J, Miranker D (2008) Translating sql applications to the semantic web. In: International Conference on Database and Expert Systems Applications. Springer, pp 450–464
- Tran T, Ladwig G, Rudolph S (2009) Istore: efficient rdf data management using structure indexes for general graph structured data. Institute AIFB, Karlsruhe Institute of Technology
- Trinkunas J, Vasilecas O (2007) Building ontologies from relational databases using reverse engineering methods. In: Proceedings of the 2007 international conference on Computer systems and technologies. pp 1–6
- Upadhyaya SR, Kumar PS (2005a) ERONTO: a tool for extracting ontologies from extended E/R diagrams. In: Proceedings of the 2005 ACM symposium on Applied computing. pp 666–670
- Upadhyaya SR, Kumar PS (2005b) ERONTO: a tool for extracting ontologies from extended E/R diagrams. In: Proceedings of the 2005 ACM symposium on Applied computing. ACM, pp 666–670
- Uschold M, King M (1995) Towards a methodology for building ontologies. Citeseer
- Van Assem M, Gangemi A, Schreiber G (2006a) Conversion of WordNet to a standard RDF/OWL representation. In: LREC. pp 237–242
- Van Assem M, Malaisé V, Miles A, Schreiber G (2006b) A method to convert thesauri to SKOS. In: European Semantic Web Conference. Springer, pp 95–109
- Villazón-Terrazas BM (2012) A method for reusing and re-engineering non-ontological resources for building ontologies. IOS Press
- Voit MM, Paulheim H (2021) Bias in Knowledge Graphs—an Empirical Study with Movie Recommendation and Different Language Editions of DBpedia. arXiv preprint arXiv:210500674
- Weiss C, Karras P, Bernstein A (2008) Hexastore: sextuple indexing for semantic web data management. Proceedings of the VLDB Endowment 1:1008–1019
- Wielinga BJ, Schreiber AT, Wielemaker J, Sandberg JAC (2001) From thesaurus to ontology. In: Proceedings of the 1st international conference on Knowledge capture. pp 194–201

- Wilkinson K, Sayers C, Kuno H, Reynolds D (2003) Efficient RDF storage and retrieval in Jena2. In: Proceedings of the First International Conference on Semantic Web and Databases. Citeseer, pp 120–139
- Wood D, Gearon P, Adams T (2005) Kowari: A platform for semantic web storage and analysis. In: XTech 2005 Conference. pp 05–0402
- Wood D, Zaidman M, Ruth L, Hausenblas M (2014) Linked Data. Manning publications co.
- Xu Z, Cao X, Dong Y, Su W (2004a) Formal approach and automated tool for translating ER schemata into OWL ontologies. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, pp 464–475
- Xu Z, Cao X, Dong Y, Su W (2004b) Formal approach and automated tool for translating ER schemata into OWL ontologies. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, pp 464–475
- Yang S, Wei R (2020) Semantic Interoperability Through a Novel Cross-Context Tabular Document Representation Approach for Smart Cities. IEEE Access 8:70676–70692
- Yiqing L, Lu L, Chen L (2012) Automatic learning ontology from relational schema. In: 2012 IEEE Symposium on Robotics and Applications (ISRA). IEEE, pp 592–595
- Yu L (2011a) A developer's guide to the semantic Web. Springer Science & Business Media
- Yu L (2011b) A developer's guide to the semantic Web. Springer Science & Business Media
- Yu L (2011c) A developer's guide to the semantic Web. Springer Science & Business Media
- Zhang G, Jia S (2009a) Ontology-based knowledge extraction for relational database schema. In: 2009 Second International Symposium on Electronic Commerce and Security. IEEE, pp 585–589
- Zhang G, Jia S (2009b) Ontology-based knowledge extraction for relational database schema. In: Electronic Commerce and Security, 2009. ISECS'09. Second International Symposium on. IEEE, pp 585–589
- Zhang H, Diao X, Yuan Z, et al (2012) EVis: a system for extracting and visualizing ontologies from databases with web interfaces. In: 2012 Fourth International Symposium on Information Science and Engineering. IEEE, pp 408–411
- Zhang H, Li Y-F, Tan HBK (2010) Measuring design complexity of semantic web ontologies. Journal of Systems and Software 83:803–814
- Zhang L, Li J (2011) Automatic generation of ontology based on database. Journal of Computational Information Systems 7:1148–1154

Zhang Z, Miller JA (2005) Ontology query languages for the semantic web: A performance evaluation. PhD Thesis, Citeseer

Zhe Y, Zhang D, Chuan YE (2006) Evaluation metrics for ontology complexity and evolution analysis. In: 2006 IEEE International Conference on e-Business Engineering (ICEBE'06). IEEE, pp 162–170

Zhou S, Ling H, Han M, Zhang H (2010a) Ontology Generator from Relational Database Based on Jena. Computer and Information Science 3:263–267

Zhou S, Meng G, Ling H (2010b) Ontologies acquisition from relational databases. Computer and Information Science 3:185

Zou X (2020) A survey on application of knowledge graph. In: Journal of Physics: Conference Series. IOP Publishing, p 012016

R2RML: RDB to RDF Mapping Language - Google Scholar.
https://scholar.google.com/scholar?q=R2RML%3A+RDB+to+RDF+Mapping+Language&hl=en&as_sdt=0%2C5&as_ylo=2012&as_yhi=2012. Accessed 13 Apr 2020