

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (Toulouse INP)

Discipline ou spécialité :

Informatique

Présentée et soutenue par :

Mme MAROUA MASMOUDI

le mercredi 1 juillet 2020

Titre :

Knowledge hypergraph based-approach for multi-source data integration and querying: Application for Earth Observation domain

---

Ecole doctorale :

Systèmes (EDSYS)

Unité de recherche :

Laboratoire de Génie de Productions de l'ENIT (E.N.I.T-L.G.P.)

Directeur(s) de Thèse :

M. BERNARD ARCHIMEDE

MME HAJER BAAZAOUI ZGHAL

Rapporteurs :

M. ERNESTO EXPOSITO, UNIVERSITE DE PAU ET DES PAYS DE L ADOUR

M. FAIEZ GARGOURI, UNIVERSITÉ DE SFAX

Membre(s) du jury :

M. KHALIL DRIRA, LAAS TOULOUSE, Président

M. BERNARD ARCHIMEDE, ECOLE NATIONALE D'INGENIEUR DE TARBES, Membre

Mme FATEN CHAIEB, UNIVERSITE DE CARTHAGE, Membre

Mme GENOVEVA VARGAS, UNIVERSITE GRENOBLE ALPES, Membre

Mme HAJER BAAZAOUI ZGHAL, UNIVERSITE DE MANOUBA, Membre

Abstract — Early warning against natural disasters to save lives and decrease damages has drawn increasing interest to develop systems that observe, monitor, and assess the changes in the environment. Over the last years, numerous environmental monitoring systems and Earth Observation (EO) programs were implemented. Nevertheless, these systems generate a large amount of EO data while using different vocabularies and different conceptual schemas. Accordingly, data resides in many siloed systems and are mainly untapped for integrated operations, insights, and decision making situations. To overcome the insufficient exploitation of EO data, a data integration system is crucial to break down data silos and create a common information space where data will be semantically linked. Within this context, we propose a semantic data integration and querying approach, which aims to semantically integrate EO data and provide an enhanced query processing in terms of accuracy, completeness, and semantic richness of response. To do so, we defined three main objectives. The first objective is to capture the knowledge of the environmental monitoring domain. To do so, we propose MEMOn, a domain ontology that provides a common vocabulary of the environmental monitoring domain in order to support the semantic interoperability of heterogeneous EO data. While creating MEMOn, we adopted a development methodology, including three fundamental principles. First, we used a modularization approach. The idea is to create separate modules, one for each context of the environment domain in order to ensure the clarity of the global ontology's structure and guarantee the reusability of each module separately. Second, we used the upper-level ontology Basic Formal Ontology and the mid-level ontologies, the Common Core ontologies, to facilitate the integration of the ontological modules in order to build the global one. Third, we reused existing domain ontologies such as ENVO and SSN, to avoid creating the ontology from scratch, and this can improve its quality since the reused components have already been evaluated. MEMOn is then evaluated using real use case studies, according to the Sahara and Sahel Observatory experts' requirements. The second objective of this work is to break down the data silos and provide a common environmental information space. Accordingly, we propose a knowledge hypergraph-based data integration approach to provide experts and software agents with a virtual integrated and linked view of data. This approach generates RML mappings between the developed ontology and metadata and then creates a knowledge hypergraph that semantically links these mappings to identify more complex relationships across data sources. One of the strengths of the proposed approach is it goes beyond the process of combining data retrieved from multiple and independent sources and allows the virtual data integration in a highly semantic and expressive way, using hypergraphs. The third objective of this thesis concerns the enhancement of query processing in terms of accuracy, completeness, and semantic richness of response in order to adapt the returned results and make them more relevant and richer in terms of relationships. Accordingly, we propose a knowledge-hypergraph based query processing that improves the selection of sources contributing to the final result of an input query. Indeed, the proposed approach moves beyond the discovery of simple one-to-one equivalence matches and relies on the identification of more complex relationships across data sources by referring to the knowledge hypergraph. This enhancement significantly showcases the increasing of answer completeness and semantic richness. The proposed approach was implemented in an open-source tool and has proved its effectiveness through a real use case in the environmental monitoring domain.

Résumé — Les dégâts humains et matériels engendrés par les catastrophes naturelles, avaient suscité un intérêt grandissant pour le développement des systèmes d'observation et de surveillance de l'environnement, sans pour autant mettre en exergue, la collaboration et l'échange comme principal point d'une efficace prévention des catastrophes. De tels systèmes génèrent des données hétérogènes et cloisonnées dans des silos. A défaut d'une vision globale des données disponibles, les experts éprouvent des difficultés à accéder, manipuler et comprendre ces données multi-source. Afin de remédier à cette insuffisance d'exploitation, un système d'intégration de données est essentiel pour briser les silos de données et créer un espace commun d'information où les données seront liées sémantiquement. C'est dans cet ordre d'idées que nous proposons une approche sémantique d'intégration et d'interrogation des données multi-sources. Pour ce faire, nous avons défini trois principaux objectifs. Le premier objectif est de formaliser les connaissances liées au domaine de l'environnement afin d'assurer une interopérabilité sémantique entre les données multi-source. Ainsi, nous avons proposé MEMOn, une ontologie de domaine qui fournit un vocabulaire commun couvrant le domaine de l'environnement. Nous avons adopté une méthodologie agile basée sur la modularisation, l'alignement avec une ontologie de haut niveau et la réutilisation des ontologies existantes. La modularisation consiste à développer des modules ontologiques séparés. Chaque module présente un contexte spécifique du domaine de l'environnement et ce dans le but d'assurer la clarté de la structure de l'ontologie globale. De plus, nous avons utilisé l'ontologie de haut niveau Basic Formal Ontology et les ontologies intermédiaires Common Core Ontologies afin de faciliter l'intégration des modules ontologiques développés pour créer MEMOn. Aussi, nous avons aussi réutilisé des ontologies de domaine existantes telles que ENVO et SSN afin d'éviter de créer notre ontologie à partir de zéro. MEMOn est ensuite évaluée à l'aide de cas d'utilisation réelles et conformément aux exigences des experts. Le deuxième objectif de ce travail est de briser les silos de données et de fournir un espace commun d'information sur l'environnement où les données pourraient être liées sémantiquement. En conséquence, nous proposons une approche sémantique d'intégration virtuelle des données basée sur l'hypergraphe afin de fournir aux experts une vue intégrée et liée des données. L'approche consisite à génrer des mappings RML entre l'ontologie et les métadonnées et à créer ensuite un hypergraphe de connaissances qui relie sémantiquement ces mappings afin d'identifier des relations plus complexes entre les données. Un des atouts de l'approche proposée est qu'elle va au-delà du processus de combinaison de données extraites de sources indépendantes pour assurer une intégration de données hautement sémantique et expressive. Le troisième objectif de cette thèse concerne l'amélioration du traitement des requêtes en termes de précision et de complétude des résultats afin d'adapter les résultats renvoyés et les rendre plus pertinents et plus riches termes de relations. En conséquence, nous avons développé une approche de traitement des requêtes basée sur l'hypergraphe de connaissances qui améliore la tâche de sélection des sources contribuant au résultat final d'une requête SPARQL saisie. En effet, l'approche proposée transcende la simple découverte de correspondances entre la requête et les schémas de sources et assure l'identification de correspondances plus complexes avec les sources de données en se référant à l'hypergraphe de connaissances. Sur la base de ces résultats, d'autres étapes du traitement de la requête, y compris la réécriture de la requête et l'évaluation de la requête, sont effectuées. Notre approche est concrétisée par le développement d'un outil dont l'efficacité a été prouvée moyennant l'évaluation d'un cas réel.

About the author

Maroua MASMOUDI

Address : 47 Avenue d'Azereix, 65000 Tarbes, France.
Email : maroua.masmoudi@enit.fr/marwa.masmoudi17@gmail.com
Born: October 17, 1990 – Tunisia

Education

2017 PhD student at the National School of Engineers of Tarbes (ENIT) and the National School of Computer Sciences Engineering of Tunisia (ENSI)
2015 Working as a Java developer in TuniTeam society, Sfax, Tunisia
2014 Engineer Degree in computer sciences, ENSI, University of Manouba, Tunisia
2011 Admission to the "Concours préparatoire aux écoles d'ingénieurs" (rank 142/1500)
2009 Scientific Baccalaureate Degree, Sfax Tunisia

Publications

- Masmoudi, M., Lamine, S. B. A. B., Zghal, H. B., Karray, M. H., and Archimède, B. (2020). Knowledge hypergraph-based approach for data integration and querying: Application to Earth Observation. Future Generation Computer Systems. (IF:5.7) Under revision.

- Masmoudi, M., Karray, M. H., Lamine, S. B. A. B., Zghal, H. B., and Archimede, B. (2020). MEMOn: Modular Environmental Monitoring Ontology to link heterogeneous Earth observed data. Environmental Modelling and Software, 124, 104581. (IF: 4.55). Published.

- Masmoudi, M., Karray, M. H., Lamine, S. B. A. B., Zghal, H. B., and Archimede, B. (2019, November). DISERTO: Semantics-Based Tool for Automatic and Virtual Data Integration. In 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA) (pp. 1-8). IEEE. Published.

- Masmoudi, M., Karray, M. H., Lamine, S. B. A. B., Zghal, H. B., Guegan, C. G, Archimede, B. Vers une plateforme sémantique pour l'intégration des données massives appliquée à la surveillance environnementale. Atelier IN-OVIVE-5ème édition, IC 2019. Published.

- Masmoudi, M., Taktak, H., Lamine, S. B. A. B., Boukadi, K., Karray, M. H., Zghal, H. B., Archimede, B., Mrissa, M., and Guegan, C. G. (2018, November). PREDICAT: a semantic service-oriented platform for data interoperability and linking in earth observation and disaster prediction. In 2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA) (pp. 194-201). IEEE. Published.

- Masmoudi, M., Lamine, S. B. A. B., Zghal, H. B., Karray, M. H., and Archimède, B. (2018). An ontology-based monitoring system for multi-source environmental observations. In the 22nd International Conference on Knowledge Based and Intelligent information and Engineering Systems (KES). Procedia Computer Science, 126, 1865-1874. Published.

To my parents (this work is the fruit of your support).

To Mahmoud (for the love you give me and your patience).

Aknowledgments

Accomplishing this thesis research work has been an exciting and challenging journey from beginning to end, with ups and downs and twists along the way. I express my sincere appreciation to those who have contributed to this thesis and supported me during this amazing journey.

First, I am deeply grateful to all members of the jury. Specifically, I would express my sincere thanks to Faiez Gargouri and Ernesto Exposito for kindly accepting to review the manuscript and also to Genoveva Vargas Solar, Faten Chaieb and Khalil Drira for agreeing to participate in the defense of this thesis.

I express my sincere gratitude to Madame Hajer Baazaoui, my doctoral advisor, for allowing me to conduct this research under her auspices. I am especially grateful for providing me with the continuous input and feedback needed. Her valuable suggestions, comments and guidance encourage me to learn more day by day.

I would like to thank Bernard Archimede, my second doctoral advisor, for encouraging my research and for sharing his knowledge and enthusiasm. His deep insights helped me at various stages of my research.

I am extremely greatful to Sana Ben Abdallah, as my academic supervisor, for sharing her ideas and helpful suggestions for my work, for the fruitful discussions and the inspiration for improvements. Thank you to be as a sister. Your support has been the most profitable experience for me.

A special thanks to my second academic supervisor, Hedi Karray, for his supervision and support. The fraternal kindness, good advice, and backing that you gave, especially in difficult times, has motivated me to work harder, strive towards my goal, and bring this research to a successful conclusion. I really appreciate your willingness to help and meet me whenever I need some clarification. I learned a lot from you.

Many thanks to all the research scholars and colleagues, especially Cendrella Chahine, Foued Abroug, Nesrine Jlassi, and Moncef souilah for their support, their help and suggestions.

I also thank all the departmental staff of both ENIT and ENSI for helping me during these years of my study both academically and officially.

This journey would not have been possible without the dedicated support of my family, who I shall now thank as best as I am able.

I would like to thank my family-in-law for their support, their comprehension. Many thanks to my sister RIM for her support and her psychological counselling, to my brother-in-law Zoubeir, and my nephews Youssef and Yacine. I am also thankful to my sister, Nour,

for her encouragements, her support and for being such a helpful and wonderful person in my life.

Words cannot express the feelings I have for my parents for their constant unconditional support both in my study and my career as well. I would not be here if it wasn't because of you. Thanks Dad for being a constant source of strength and inspiration to me. Thanks Mom, your prayer for me was what sustained me this far and put me through from many difficulties that I am facing for all these years.

And last but not least, a heartfelt thanks to my loving husband, Mahmoud, for being my trusted confidant, providing me with all the patience, love, inspiration, and support needed to endure and finish this project. We have faced many challenges, traveled many miles and shared many stories together. This work belongs to you. I will love you, forever and for always.

Contents

Introduction **1**

1 Conceptual Background **7**

1.1 Introduction 7

1.2 Semantic Web and ontologies 8

1.2.1 Semantic Web 8

1.2.2 Ontologies 8

1.2.3 SW technologies and languages 10

1.2.3.1 Resource Description Framework (RDF) 11

1.2.3.2 Web Ontology Language (OWL) 12

1.2.3.3 SPARQL 13

1.2.3.4 SWRL 14

1.2.3.5 Ontology programming tools 15

1.2.4 Modular ontology 15

1.3 Data integration 16

1.3.1 Data integration architectures 17

1.3.2 Semantic data integration with ontologies 21

1.3.3 Data integration techniques 22

1.3.3.1 Schema matching 22

1.3.3.2 Schema mapping 22

1.4 Query processing 25

1.4.1 Query processing types 25

1.4.2 Distributed query processing principles 26

1.5 Hypergraphs 28

1.6 Conclusion 29

2 Context and State of the art **31**

2.1 Introduction 31

2.2 Scope of this research 32

2.2.1 Applicative context 32

2.2.2 Data challenges 33

2.2.3 Scope: PREDICAT project 35

2.2.4 Focus of this thesis 38

2.3 State of the art of ontologies related to EO 39

2.3.1 Sensor ontologies 39

2.3.2 Observation and measurement ontologies 40

2.3.3 Environmental monitoring ontologies 41

2.3.4 Synthesis 45

2.4 State of the art of semantic data integration approaches 45

2.4.1 Materialized approaches 45

2.4.2 Virtual approaches 47

2.4.2.1 Mediator-wrapper-based systems 47

2.4.2.2 Federated systems 49

2.4.3 Synthesis 50

2.5 Discussion and orientations 55

2.6 Conclusion 57

3 MEMOn: Modular Environmental Monitoring Ontology **59**

3.1 Introduction 59

3.2 Ontology development methodology 59

3.3 MEMOn development process 60

3.3.1 Exploration phase . . . 61
3.3.1.1 MEMOn objectives . . . 61
3.3.1.2 MEMOn requirements . . . 62
3.3.2 Planning phase . . . 62
3.3.3 Module development phase . . . 65
3.3.3.1 Basic Formal Ontology (BFO) . . . 66
3.3.3.2 The Common Core Ontologies (CCO) . . . 66
3.3.3.3 Semi-formal MEMOn modules building . . . 69
3.3.3.4 Module formalization and evaluation . . . 81
3.3.4 Release phase . . . 86
3.3.4.1 MEMOn implementation and enrichment . . . 86
3.3.4.2 MEMOn evaluation . . . 87
3.4 Conclusion . . . 95

4 Knowledge hypergraph-based approach 97

4.1 Introduction . . . 97
4.2 Motivating example . . . 98
4.3 Architecture of the proposed approach . . . 100
4.4 Hypergraph-based virtual data integration . . . 101
4.4.1 Semantic annotation . . . 102
4.4.2 RML mappings generation . . . 105
4.4.3 Knowledge hypergraph building . . . 107
4.5 Hypergraph-based query processing . . . 111
4.5.1 Query parsing . . . 112
4.5.2 Hypernodes selection . . . 114
4.5.3 SPARQL sub-queries rewriting . . . 115
4.5.4 Data consolidation and query execution . . . 115

4.6 Conclusion 118

5 Experimentation and evaluation **121**

5.1 Introduction 121

5.2 Onto-KIT tool 121

5.2.1 DISERTO tool 123

5.2.2 HyQ tool 129

5.3 Performance evaluation 132

5.3.1 Experimental setup 132

5.3.2 Experiment 1: DISERTO: semantic annotation evaluation 133

5.3.3 Experiment 2: HyQ evaluation 134

5.3.4 Experiment 3: Comparison with Karma 135

5.4 Discussion 137

5.5 Conclusion 139

6 Conclusions and perspectives **141**

6.1 Summary 141

6.2 Limitations 144

6.3 Perspectives 145

6.3.1 Ongoing directions 145

6.3.2 Potential directions 146

Bibliography **148**

Bibliography **149**

List of Figures

1 Mind Map of our research methodology. 5

1.1 Tim Berners-Lee's Semantic Web Stack (Berners-Lee et al., 2001). 9

1.2 Ontology classification based on domain scope (Falquet et al., 2011). 10

1.3 Graph representation of an RDF triple. 11

1.4 Data integration system. 17

1.5 Materialized data integration system. 18

1.6 Mediator-wrapper-based data integration system. 19

1.7 SPARQL endpoint federation system. 20

1.8 RML overview. 24

1.9 Distributed Query Processing steps. 27

1.10 A generalized hypergraph example. 29

2.1 PREDICAT architecture. 36

2.2 Focus of this thesis regarding PREDICAT. 38

2.3 (a) Traditional virtual data integration framework (Zidane et al., 2018). b) Our orientation to VKHG -based data integration framework. 56

2.4 Proposed hybrid query processing architecture. 57

3.1 The adapted AOM development life-cycle. 61

3.2 Environmental monitoring domain' expert stories. 63

3.3 A fragment of the BFO's class hierarchy. 66

3.4 CCO modules hierarchy. 68

3.5 Import and reuse of ontologies in MEMOn modules. 70

3.6 Partial view of the sensor&sensing module. 72

3.7 Partial view of the observation&measurement module. 73

3.8 Partial view of the environmental material module. 75

3.9 Partial view of the environmental process module. 77

3.10 Partial view of the natural disaster module. 77

3.11 Partial view of the infrastructure module. 79

3.12 Partial view of the geospatial module. 80

3.13 Partial view of the temporal module. 81

3.14 Partial view of MEMOn modules. 87

3.15 SPARQL query and results of the CQ1. 88

3.16 SPARQL query and results of the CQ2. 89

3.17 SPARQL query and results of the CQ3. 89

3.18 The part of MEMOn used to define a five-category hurricane. Each MEMOn module is marked with a unique color. XML schema datatypes are shown in blue. 90

3.19 SPARQL query and the result of the 5-Category hurricane. 90

3.20 A global schema of the use case deployment. 91

3.21 Storm events data extracted from the NOAA web service. 92

3.22 Snapshot of Karma for the data mapping. 93

3.23 SPARQL query script and the results of the natural disasters between 2016 and 2018. 93

3.24 SPARQL query and results of the flood disasters and their corresponding precipitation values. 94

3.25 Part of the knowledge graph linking multi-source data with MEMOn. 94

4.1 Partial view of the metadata from the raster image originating from OSS. . . 98

4.2 Partial view of data in JSON format from AerisWeather API. 98

4.3 Partial view of data about Hourly Precipitation from the NOAA. 99

4.4 Motivating example. (a) SPARQL query over data sources; (b) heterogeneous data sources. 99

4.5 Architecture of the proposed knowledge hypergraph-based data integration and querying approach. 101

4.6 The whole process of the hypergraph-based virtual data integration phase. . . 102

4.7 (a) An RML mapping graph (`RML_G`). (b) A semantic mapping view over `RML_G` 108

4.8 Observation-oriented sub-hypergraph describing the relations among observations. 110

4.9 The process of the virtual integration from data to knowledge hypergraph building. 111

4.10 The entire hypergraph-based query processing. 112

4.11 The corresponding graph pattern and schema pattern to the query in Listing 4.1 113

5.1 Onto-KIT GUI. 122

5.2 DISERTO GUI. 123

5.3 Activity diagram of DISERTO. 124

5.4 The mapping view hypernode "`H_001`". 126

5.5 The relations between "`H_001`" and corresponding hyperedges. 126

5.6 A partial view of the precipitation-oriented sub-hypergraph. 128

5.7 A partial view of the knowledge hypergraph. 128

5.8 HyQ GUI. 129

5.9 Sequence diagram of the query processing. 130

5.10 (a) Example of SPARQL query. (b) The corresponding schema pattern. (c) The query used to interrogate S_RDFStore 130

5.11 (a) The query used to interrogate the knowledge hypergraph. (b) Results. . . 131

5.12 Performance variation in terms of precision, recall and F1-measure (a) Only with domain ontology exploitation. (b) with ontology, thesaurus, and annotations exploitation. 133

5.13 Performance variation of HyQ in terms of execution time. 135

5.14 Performance variation in terms of completeness and cardinality of returned results. 137

5.15 Performance variation in terms of execution time and cardinality. 138

List of Tables

2.1 Comparison of existing ontologies related to EO. 44

2.2 Comparison of existing semantic data integration approaches. 51

2.2 Comparison of existing semantic data integration approaches. 52

2.2 Comparison of existing semantic data integration approaches. 53

3.1 Competency questions examples. 64

3.2 BFO classes definitions. 67

3.3 CQs related to the sensor&sensing module. 70

3.4 CQs related to the observation & measurement module. 73

3.5 CQs related to the environmental material module. 74

3.6 CQs related to the environmental process module. 76

3.7 CQs related to the natural disaster module. 76

3.8 CQs related to the infrastructure module. 78

3.9 CQs related to the geospatial module. 79

3.10 CQs related to the temporal module. 81

3.11 MEMOn Metrics' results. 85

3.12 Classes and relations of MEMOn 86

3.13 Some examples of MEMOn's SWRL rules. 88

List of acronyms

SW	Semantic Web
MEMOn	Modular Environmental Monitoring Ontology
PREDICAT	PREDIct natural CATastrophes
BFO	Basic Formal Ontology
CCO	Common Core Ontologies
RO	Relations Ontology
ENVO	ENVironment Ontology
SOSA	Sensor, Observation, Sample, and Actuator
SSN	Semantic Sensor Network ontology
UMO	Unit of Measure Ontology
EO	Event Ontology
GEO	GEospatial Ontology
TO	Time Ontology
SPARQL	Simple Protocol and RDF Query Language
OWL	Web Ontology Language
SWRL	Semantic Web Rule Language
CQs	Competency Questions
AOM	Agile methodology for developing Ontology Modules
OBDA	Ontology-Based Data Integration
OBDI	Ontology-Based Data Access
OSS	Sahara and Sahel Observatory
NOAA	National Oceanic and Atmospheric Administration
RDF	Resource Description Framework
RML	RDF Mapping Language
GUI	Guided User Interface
Onto-KIT	Ontology-based Knowledge hyergraph data Integration and querying Tool
DISERTO	Data Integration Semantic hypERgraph Tool
HyQ	Hypergraph-based Querying
VoID	Vocabulary of Interlinked Datasets
CSV	Comma Separated Values

Introduction

Context and motivations

In recent years, the Earth has undergone rapid climate change and natural activities, which are believed to increase the number of natural disasters, such as storms, floods, earthquakes, and hurricanes. These disasters have dramatically influenced not only the natural environment but also human life. Consequently, research communities have given great importance to the development and implementation of Earth Observation (EO) systems (such as sensors and satellite platforms (Sentinel, 2000)) and environmental monitoring programs (such as Copernicus (Copernicus, 2014) and SERVIR Global (SERVIR, 2005)). Along with the increased number of monitoring solutions, a multitude of heterogeneous EO data is generated. However, these data are often hidden in isolated silos, maintained in legacy systems and sometimes are not digitally available, or agreement to specific laws and regulations is required to access them. Issues of managing data in terms of copyright and licensing, pricing, and data rights, though changing fast, are commonly difficult and still limit open access to EO data.

Accordingly, the exploitation of EO data is limited, and experts' involvement is still required to scout for data that are needed for integrated studies. Undeniably, we have not reached a level where data are interoperable and linked so that experts can reuse them soundly. We are still far away from the vision of common environmental information space (Athanasiadis, 2015). Several environmental events serve as examples of how the absence of linked observed data hinders the anticipation and the understanding of natural phenomena. One of the most known disasters' examples is Hurricane Irma, which occurred across the Caribbean in 2017 (IRMA, 2017). Indeed, the lack of a common environmental information space between the national hurricane center of NOAA (NOAA, 2014), that monitors the Atlantic basin and NASA (Emmons et al., 2007) that observes the African Sahara Desert has hampered the prediction of this devastating disaster power and consequently delayed the governments' alerting so that they could have taken more preventive actions.

Our purpose is to break down the data silos to provide what we call a global information view, where different EO systems will have not only unhampered and uniform access to the available data but also be able to interpret and use them. This global information view allows the data sources to speak the same language and to link information so that domain experts could transform them into actionable knowledge. We refer here to a knowledge graph (KG) (Ehrlinger and Wöß, 2016). A KG is defined as a multi-relational graph composed of entities and relationships between them. With this KG, experts can look at all of this data and try to find meaning out of its correlations to understand natural phenomena and make the right decisions about disasters' risk preventions.

A global information view is further challenged by semantic data integration. Semantic

data integration goes beyond the data integration process of combining data retrieved from independent sources to provide an integrated and interoperable structure (Lenzerini, 2002). Indeed, semantic data integration is the process of combining and consolidating disparate data into meaningful and valuable information by creating links between them in order to create a richer global view (Cheatham and Pesquita, 2017). Unfortunately, several challenges are confronted by semantic data integration. One of the most obvious is data heterogeneity, which can appear at different levels. In fact, EO data are generated in different formats (databases, CSV files, Raster images, etc.). Besides, each data source has its own and different data model or schema. Furthermore, data is semantically heterogeneous (synonymy, polysemy, etc.). Each source offers data or semantic models encoding domain knowledge that resides in the experts' minds. Thus, EO experts or data analysts need to establish contact with original data sources and model producers to understand and use them properly. For example, the Observatory of Sahara and Sahel (OSS) (OSS, 2010) may use the word "rainfall" for the same real-world feature that usually refers to "precipitation" in other sources, including NOAA. This heterogeneity of terms complicates the work of EO experts and software agents who should be familiar with the vocabulary used in each source to understand data. Accordingly, with this extensive heterogeneity of EO data, it is becoming increasingly difficult for domain experts to understand natural phenomena and reduce the adverse effects of climate change.

Another fundamental challenge appears when developing an integration system and which is the processing and answering of users' queries. Several systems were developed to improve and optimize query processing in terms of accuracy and runtime. These works focused on generating methods to enhance the execution of the different query processing steps: source selection, query planning, query evaluation, etc. However, devising source selection approaches has not received much attention, despite the importance of this task in the query processing. Source selection enables to identify the relevant data sources to an input user's query. This latter typically represents an exact expression of the user's needs. However, because of the dynamic nature of the data integration context and the abundance of data sources, users may not know the data sources they questioned, nor their content. Due to the non-transparency of sources' contents, it can be possible that a relevant source does not contribute to the result of a query. Accordingly, the queries reflect no more a need that must be satisfied but an intention that must be extended according to data sources. Consequently, a user, with the intention of satisfying an information need, may have to reformulate the query several times and sift through many results until a satisfactory one. For instance, a traditional query processing engine running a query that asks for atmospheric temperature in a specific country, represented by its name, will only extract data from sources representing countries by the name. Data sources that describe the requested country by its geographic coordinates will not contribute to the result, although they contain relevant data. Clearly, query processing needs to reach every possible source to obtain all possible answers. Thus, the need to move beyond the discovery of simple one-to-one equivalence matches to the identification of more complex relationships across datasets.

Accordingly, new challenges arise to enhance the processing of queries in terms of completeness and establish an appropriate method for enriching the returned results to a query and make them more precise and more relevant. Representing the relationships between earth

observations in the global information view so as to highlight the relationships between data sources allows the query processing to return more focused, relevant, and hopefully optimal answers. Therefore, enhancing the query processing in terms of response completeness requires finding solutions to two interdependent issues: improving the modeling of the global information view, and improving the selection of relevant data sources against the user's query.

Thesis objectives

In this thesis, we tackle the challenges of enabling semantic data integration and querying over heterogeneous EO data sources. The aim is to improve the semantic integration and linking among the multi-source data and to ensure an enhanced information extraction in terms of accuracy, completeness, and relationships richness. According to those challenges, the main objectives of this thesis are summarized in the following:

- To have a better overview and mutual understanding of the environmental monitoring domain and to ensure information exchange among experts and software agents by representing and enriching the knowledge existing in EO data sources.
- To semantically link data so as to build a huge knowledge graph that, covering the environmental monitoring domain, provides a global information view that takes full advantage of heterogeneous data.
- To propose an enhanced query processing approach that allows to transparently query distributed data sources and cover a broadening spectrum of user queries' answers while taking into account the results accuracy, completeness, and semantic richness challenges.

Research questions

To achieve the aforementioned objectives, this thesis will address the following main research questions:

- **RQ1:** How to formalize the knowledge related to the environmental monitoring domain?
- **RQ2:** How to break down the data silos and deal with interoperability issues? This research question is divided into several sub-questions:
  - RQ2a: How to address the semantic, syntactic, and schematic heterogeneity issues that hamper information exchange/interoperability among EO data sources?
  - RQ2b: How to allow highly semantic expressive linked data in the context of spatiotemporal data?
  - RQ2c: How to ensure a semantic global view of information?

- **RQ3:** How to improve the query engine performance in terms of results accuracy, completeness, and relationship richness?

Contributions

Our work consists of four main contributions. First, we propose and develop an ontology that formalizes the environmental monitoring domain. It gives an overview of the different components/contexts which contribute to an environmental monitoring system, including environmental conditions (e.g., environmental phenomena), observing conditions (e.g., sensors and measurements), infrastructure, and spatiotemporal context. The ontology also describes relationships between the four components. Thus, while conceptualizing the ontology, we adopt the principle of modularization. The idea is to create separate modules, one for each context of the environment domain in order to ensure the clarity of the ontology's structure and guarantee ontology's evolution and maintenance. Furthermore, we reuse an upper-level, mid-level, and domain ontologies to promote the semantic interoperability among the proposed ontological modules and existing ontologies in the same domain. The proposed ontology aims to provide a common knowledge representation for the environmental monitoring domain and to facilitate semantic linking of data from different sources through a knowledge graph.

Second, we exploit the proposed ontology to integrate the multi-source and heterogeneous earth observations in order to provide experts and software agents with an integrated and linked view of data. Accordingly, we propose a semantic data integration approach that virtually integrates EO data by maintaining it in their sources and generates mappings between data and the ontology in order to build a global information view. In addition, to highlight the relationships between earth observations, we propose to use hypergraphs to model the global information view. Hypergraphs are able to describe complex relationships between earth observations because of their better expressive capabilities. Accordingly, the output of the data integration approach is a knowledge hypergraph.

Then, we exploit the generated knowledge hypergraph to retrieve and adapt the returned results to a user query and make them more relevant and semantically richer. Therefore, we propose an enhanced query processing approach to consolidate relevant data from relevant sources in order to enhance information extraction in terms of accuracy, completeness, and semantic richness. Specifically, the source selection task of the query processing is enhanced to identify relevant sources that possibly contribute to the final result.

Finally, we propose an open-source prototype implementation of the semantic data integration and querying approach, that we evaluate in a real-world use case. Figure 1 illustrates the mind map of the research methodology.

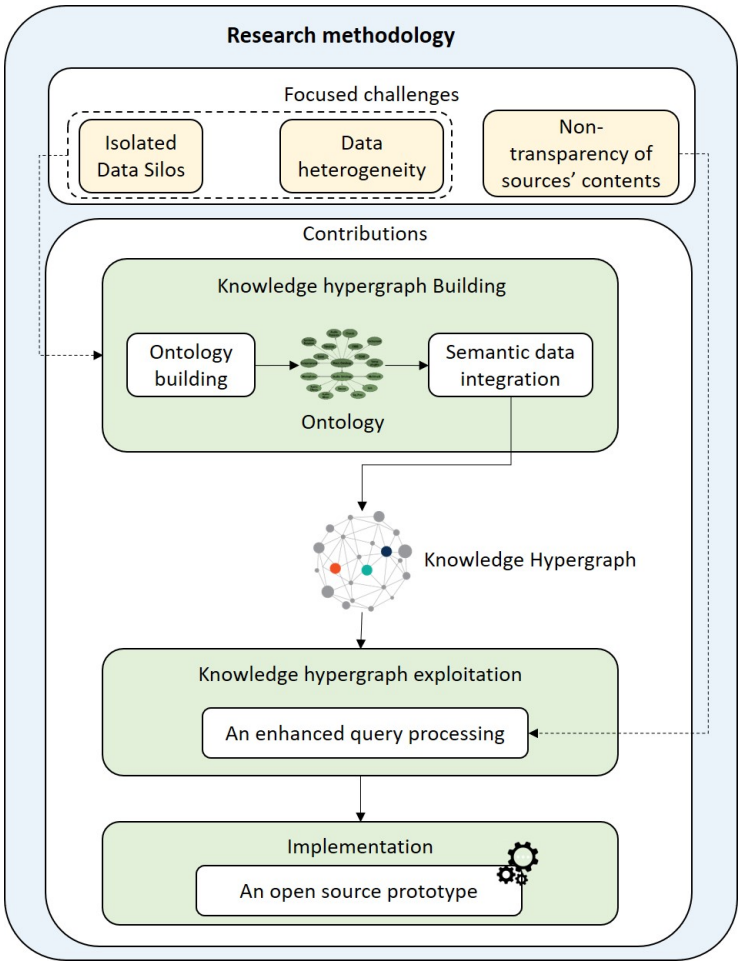


Figure 1: Mind Map of our research methodology.

Thesis outlines

Chapter 1 is devoted to providing an overview of topics related to this thesis. First, we introduce the semantic web and the basic concepts related to it. Then, we present the general principles of data integration followed by the description of the two main categories of data integration approaches; the materialized and the virtual approaches and point out their implications on the query answering process. Afterward, we remind the fundamentals of hypergraphs.

Chapter 2 is divided into two parts. The first part gives an overview of the project PREDICAT, scope of this work, by defining the context and the motivations of the project. Then, the architecture of PREDICAT platform is presented to specify the context of this thesis and better understand its positioning and its orientations. In the second part, we present the state of the art. First, we give a research overview of the existing ontologies in the earth observation and environmental monitoring domains. Then, we discuss different semantic data integration and query processing approaches and based on the identified research gaps, we

underline the challenges that should be considered in this work.

In chapter 3, we present the proposed domain ontology that semantically captures the knowledge related to the environmental monitoring domain. The ontology's development approach is detailed, following the different steps of the adopted agile ontology building methodology. Specifically, we describe how the environmental monitoring domain is conceptualized and formalized. Then, we demonstrate how the implemented ontology is evaluated through a verification step against the ontology requirements initially identified and a validation step using a real use case study.

Chapter 4 depicts the proposed semantic data integration and querying approach that promotes semantic linking of the multi-source data and improves the query processing in terms of accuracy, completeness, and semantic richness. The two-phases architecture of the proposed approach is presented. The first phase presents the hypergraph-based data integration, and the second phase presents the enhanced query processing. The mechanisms of the two phases are explained in detail.

Chapter 5 reports the experimental results to assess the performance of the proposed data integration and querying approach. First, we present Onto-KIT, the tool that implements the proposed approach. Then, we present the performed experiments and their relative results, and we discuss the impact of our approach regarding several challenges, namely schema matching accuracy, query results completeness, and semantic richness.

Finally, in chapter 6, we provide concluding remarks, the limitations of our work and suggest potential future directions.

Chapter 1

Conceptual Background

Sommaire

1.1 Introduction 7
1.2 Semantic Web and ontologies 8
1.2.1 Semantic Web 8
1.2.2 Ontologies 8
1.2.3 SW technologies and languages 10
1.2.4 Modular ontology 15
1.3 Data integration 16
1.3.1 Data integration architectures 17
1.3.2 Semantic data integration with ontologies 21
1.3.3 Data integration techniques 22
1.4 Query processing 25
1.4.1 Query processing types 25
1.4.2 Distributed query processing principles 26
1.5 Hypergraphs 28
1.6 Conclusion 29

1.1 Introduction

This chapter is intended to provide the reader with an overview of the scientific and technological context of this thesis. For a general standpoint, this work is mainly influenced by two areas of research that are frequently considered in relation to each other, namely, ontology engineering and data integration. Therefore, in this chapter, we will introduce the basic concepts of these research areas. We first describe the semantic web and its related concepts. Specifically, we remind the principles of ontologies. Then, we present the fundamentals of data integration, followed by the description of two categories of data integration approaches (materialized and virtual) and explain their implications on the query answering process. Finally, we synthesize the fundamentals of hypergraphs, which are used in the approach proposed in this work.

1.2 Semantic Web and ontologies

"The Semantic Web (SW) is an extension of the current web in which information is given well-defined meaning, better-enabling computers, and people to work in cooperation" (Berners-Lee et al., 2001). The term "Semantic Web" has been disseminated by Berners-Lee et al. as referring to a vision of an intelligent web and refers to the Worldwide Web Consortium (W3C)'s vision of the Web of linked data. The term most closely related to the development of the SW is "ontology". This is because ontologies, which provide a formal vocabulary of concepts and the axioms relating them, are used to annotate and describe the "semantics" of the data in a way meaningful for machine interpretation. Thus, in the following sections, concepts relevant to SW and ontologies are outlined.

1.2.1 Semantic Web

The Semantic Web, according to Tim Berners-Lee, is a web of data, in some way, like a global database. The SW "will bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users" (Berners-Lee et al., 2002). In addition to the classic Web of documents, W3C is helping to build a technology stack to support a Web of data. The goal of SW is to make Internet data machine-readable. To allow the encoding of semantics with the data, and enable web agents to understand these data, technologies such as Resource Description Framework (RDF) and Web Ontology Language (OWL) are used. SW technologies allow people to create data stores on the Web, build vocabularies, and write rules for handling data.

It has been widely accepted that the architecture of the SW will be based on a hierarchy of languages, each of which both exploits the features and extends the capabilities of the layers below. This has been illustrated in Tim Berners-Lee's famous "Semantic Web Stack," presented in Figure 1.1 It shows how technologies that are standardized for SW are organized to make the SW possible. It also shows how SW is an extension (not replacement) of the classical hypertext web. The stack is still evolving as the layers are concretized. The technologies from the bottom of the stack up to OWL are currently standardized and accepted to build SW applications. As one of the building blocks of Semantic Technologies, ontologies are part of the W3C standards stack for the SW. They provide users with the necessary structure to link one piece of information to other pieces of information on the Web of Linked Data.

1.2.2 Ontologies

The term "ontology" has different meanings in different contexts. In philosophy, it is a branch of metaphysics and is the study of the kinds of things that exist (Hofweber, 2014). However, in computer science, an ontology is typically defined as follows: "An ontology is an explicit specification of a conceptualization" (Gruber, 1995).

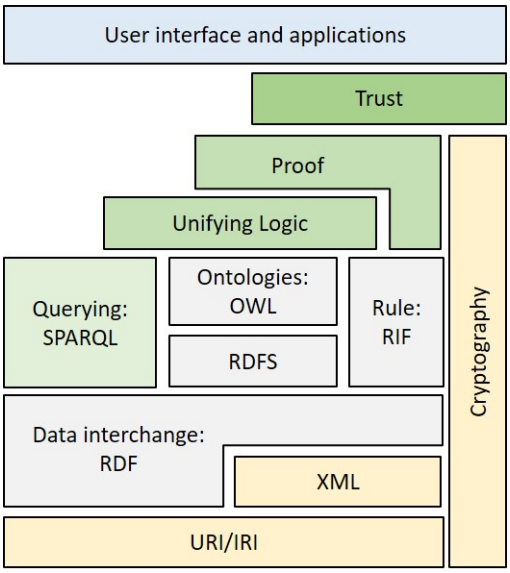


Figure 1.1: Tim Berners-Lee's Semantic Web Stack (Berners-Lee et al., 2001).

Ontologies have been widely used for knowledge representation as they provide a shared vocabulary for modeling a specific domain by capturing knowledge in a structured and formal way. According to (Maedche and Staab, 2001), an ontology is formally defined as follows:

Definition 1.2.1. An ontology O is defined as a 3-tuple: O = <C,R,A>, where:

(i) C is a set of elements called classes,

(ii) $R \subseteq C \times C$ is the set of relations between classes and contains the existing inherent hierarchical structure among the classes in C (hierarchical taxonomy),

(iii) A is the set of axioms in O.

Several classifications were presented in the literature to differentiate ontology categories. Figure 1.2 illustrates a classification based on domain scope as given in (Falquet et al., 2011).

- **Foundational, upper, or top-level ontologies** are generic ontologies that can be viewed as meta ontologies describing the high-level concepts or universals used to define other ontologies. They guarantee interoperability between domain ontologies sharing the same upper-level ontology (Noy, 2004) and facilitate the integration and knowledge reuse. The most well-known foundational ontologies are the Suggested Upper Merged Ontology (SUMO) (Pease et al., 2006), the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) (Masolo et al., 2002) and the Basic Formal Ontology (BFO) (Arp et al., 2015).

- **Mid-level ontologies** provide more concrete representations of abstract entities defined in the upper-level ontology. They serve as a bridge between abstract entities

defined in the upper-level ontology and low-level domain specified in the domain ontology.

- In contrast, **domain ontologies** are only applicable to a domain with a particular point of view. They describe the vocabulary related to a specific area of knowledge (such as medicine, environment, etc.), specializing terms introduced in the upper-level ontology.

- Finally, **application ontologies** are specializations of domain ontologies. They describe the concepts based on a particular context of the domain.

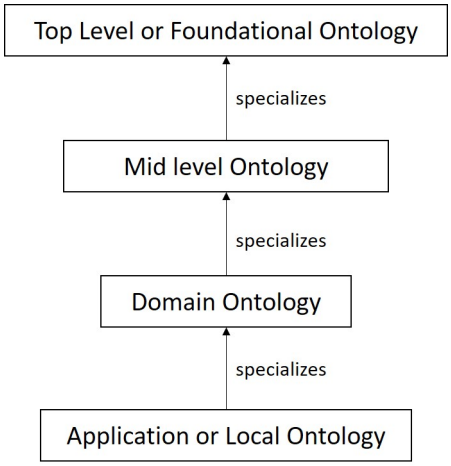


Figure 1.2: Ontology classification based on domain scope (Falquet et al., 2011).

The construction of an ontology is a task that requires knowledge from experts in the field of knowledge to be described, as well as engineering skills from the ontologists themselves. Ontology engineering is the discipline that investigates the principles, methodologies, tools, and languages for initiating, developing, and maintaining ontologies (Sure et al., 2009). To assist domain experts in building ontologies, a methodology that guides the development process is required. In fact, an ontology development methodology comprises a set of established principles, processes, practices, methods, and activities used to design, construct, evaluate, and deploy ontologies (Gaševic et al., 2009). Accordingly, different ontology development methodologies have emerged e.g. METHONTOLOGY (Fernández-López et al., 1997), NeOn (Networked Ontologies) (Suárez-Figueroa et al., 2012), AOM (Agile methodology for developing Ontology Modules) (Gobin, 2013) and OntoClean (Guarino and Welty, 2004). A synthesis of these different methodologies is presented in (Karray et al., 2012).

1.2.3 SW technologies and languages

Middle layers of the SW stack contain technologies standardized by W3C to enable building semantic web applications. In this section, we provide a brief description of some SW related technologies and languages used further in this thesis.

1.2.3.1 Resource Description Framework (RDF)

The Resource Description Framework (RDF) is a family of W3C specifications originally designed as a metadata data model (RDF, 2014). It is used as a general method for conceptual description or modeling of information, which is implemented in web resources, using a variety of syntax notations and data serialization formats. RDF is a standard model for data interchange on the Web. It has features that facilitate data merging even if the underlying schemas differ. It specifically supports the evolution of schemas over time without requiring all the data consumers to be changed. RDF extends the linking structure of the Web by using URIs (Berners-Lee, 2002) to name the relationships between things as well as the two ends of the link. This is usually referred to as a "triple". Each triple (s,p,o) consists of a subject s, a predicate p, and an object o. The predicate denotes the relationship between subject and object. Using this simple model, RDF allows structured and semi-structured data to be mixed, exposed, and shared across different applications. This linking structure forms a directed, labeled graph, where the edges represent the named link between two resources, represented by the graph nodes. This graph view is the easiest possible mental model for RDF and is often used in easy-to-understand visual explanations. Formally, an RDF triple is defined as follows:

Definition 1.2.2. RDF triple Assume there are pairwise disjoint infinite sets I, B, and L (IRIs, Blank nodes, and literals). A triple (s, p, o) $\in$ (I $\cup$ B) $\times$ I $\times$ (I $\cup$ B $\cup$ L) is called an RDF triple, where s is the subject, p the predicate, and o the object.

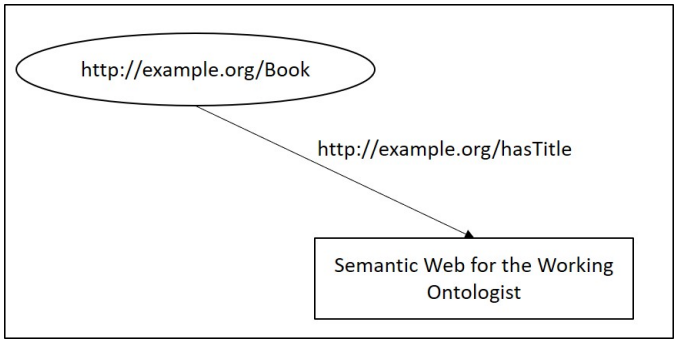


Figure 1.3: Graph representation of an RDF triple.

The example in Figure 1.3 shows a representation of an RDF triple; the statement S "The book has the title Semantic Web for the Working Ontologist" is modeled as a triple with "book" as the subject, "has title" as the predicate, and "Semantic Web for the Working Ontologist" as the object. Both the subject and the predicate are identified by a URI, while the object can be a URI or a literal value (i.e., a string or a number).

RDF triples can be written in different syntaxes such as RDF/XML, Turtle, N-triples, and JSON-LD. The graph in Figure 1.3 can be represented in Turtle as:

```
@prefix ns: <http://example.org/>.
ns:Book ns:hasTitle "Semantic Web for the Working Ontologist".
```

Listing 1.1: A statement expressed in Turtle format.

The same statement S written in RDF/XML format:

```
<rdf:RDF xmlns:ns="http://example.org/">
  <rdf:Description rdf:about="http://example.org/Book">
    <ns:hasTitle
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Semantic Web for the Working Ontologist </ns:hasTitle>
  </rdf:Description>
</rdf:RDF>
```

Listing 1.2: A statement expressed in RDF/XML format.

S written in N-Triples format:

```
<http://www.w3.org/2001/sw/RDFCore/ntriples/> <http://example.org/Book
   "> <http://example.org/hasTitle"> "Semantic Web for the Working
   Ontologist".
```

Listing 1.3: A statement expressed in N-Triples format.

N-Quads format extends N-Triples with an optional context value in the fourth position. This new value can be added to describe the context of the triple, which is called the <context>, and that becomes RDF quads instead of triples (Katib et al., 2017). The following listing illustrates the quad model corresponding to statement S with BookContext representing the context.

```
<http://example.org/Book"> <http://example.org/hasTitle"> "Semantic
   Web for the Working Ontologist". <http://example.org/BookContext>
```

Listing 1.4: A statement expressed in N-Quads format.

An RDF graph is a set of RDF triples (Klyne et al., 2009). An RDF graph is typically loaded into an RDF triple store. The RDF Triple Stores or RDF databases are specifically designed to store collections of RDF triples, to support the standard SPARQL query language, and possibly to allow some kind of inference via semantic rules. Examples of RDF databases are Jena (McBride, 2002), OpenLink Virtuoso (Erling and Mikhailov, 2009), and RDF4J (formerly OpenRDF Sesame) (RDF4J, n.d.).

1.2.3.2 Web Ontology Language (OWL)

In order to extend the limited expressiveness of RDF Schema, a more expressive Web Ontology Language (OWL) has been defined by the W3C (Antoniou and Van Harmelen, 2004). OWL is part of the W3C's Semantic Web technology stack, layered on top of RDF, and can be used to describe the set of facts for authoring ontologies which builds on RDF. OWL uses

RDF namespaces to integrate the many different and incompatible ontologies that exist. The primary goal of OWL is to be able to describe concepts and relationships as unrestricted as possible, but also allow computers to infer and reason about them.

OWL can be divided into a family of three languages; Full, DL, and Lite. OWL Full is the complete language, while the others are subsets or restrictions of it. OWL-DL, supports Description Logics, a logic family applying carefully selected restrictions on what can be expressed. OWL Lite has more limitations in order to make the creation of an ontology easier. It is used for taxonomies and simple constraints. OWL 2 is an extension and revision of OWL developed by the W3C Web Ontology Working Group and published in 2004. The RDF-based semantics can be applied to any OWL 2 Ontology, without restrictions, as any OWL 2 Ontology can be mapped to RDF.

1.2.3.3 SPARQL

SPARQL Protocol And RDF Query Language (SPARQL) is an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in RDF format (Prud'Hommeaux et al., 2007). SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. This language also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs. A SPARQL query is formally defined as:

Definition 1.2.3. A SPARQL query is a 4-tuple <GP, DS, SM, R>, where:

(i) GP is a graph pattern. Several forms of GP exist. The most used one is the basic graph pattern (BGP), which combines the triples patterns of a query. The graph pattern of a query is also called a query pattern.

(ii) DS is an RDF Dataset,

(iii) SM is a set of solution modifiers. A solution sequence modifier is one of (Order, Projection, Distinct, Offset, and Limit modifiers)

(iv) R is a result form. SPARQL provides four different forms of query: SELECT, CONSTRUCT, DESCRIBE, ASK. Among these forms, SELECT query is the most frequently used query form (Yu, 2011).

A SPARQL select query has the following general form:

- PREFIX (Namespaces prefixes).
- SELECT (Result Set).
- FROM (Data Set) the link to the URI where dataset(rdf/ttl) file resides.

- WHERE (Query Triple Pattern).
- Modifiers, example: ORDER BY.

Listing 1.5 illustrates an example of a SPARQL select query.

```
\textcolor{blue} {# prefix declarations}
PREFIX flight: <http://www.example.com#>
# result clause
SELECT DISTINCT ?destination
# dataset definition
FROM <http://www.w3.org/Flights/>
# query pattern
WHERE
{ ?f flight:DestinationCity ?destination.
?f flight:hasAirline ?airline.
?airline flight:hasName "AA";   }
# query modifiers
ORDER BY(?destination)
```

Listing 1.5: An example of a SPARQL select query.

1.2.3.4 SWRL

The Semantic Web Rule Language (SWRL) was designed to be the standard rule language of the Semantic Web (Horrocks et al., 2004). It allows users to write rules expressed in terms of OWL concepts to reason with OWL individuals. The rules can be used to infer new knowledge from existing OWL knowledge bases.

To overcome the expressiveness limits of OWL2, we use the SWRL language. An SWRL rule includes two parts and is described as "antecedent" → "consequent." This signifies that if all the conditions in the antecedent are held, then all atoms in consequent must also be held. Atom is the basic component that appears in a SWRL rule. In SWRL, properties and individuals defined in the OWL are applied in the atom clause as the attribute and the parameter of the atom, respectively. There are many sorts of atoms, but in our work, two common atoms in SWRL syntax are used in the reasoning phase of problem-solving:

- C(?x): If x is an instance of the class C or the value of its data property, then C(?x) holds.
- P(?x, ?y): If x is related to y via property P, then P(?x, ?y) holds. Here P is the property defined in the existing ontology, x and y can be variables, individuals, or the data value.

It's to note that there are also built-in functions in the SWRL syntax that are capable of describing the logical comparison relationship, including *swrlb:lessThan* and *swrlb:greaterThan*.

1.2.3.5 Ontology programming tools

Several methods and tools have been proposed for the development, the edition, the maintenance, or the evaluation of ontologies. Ontology Editors are open-source and commercial tools that assist in the development of ontologies, including the Protégé editor (Protégé, 2014). They can be applied to several stages of the ontology life cycle, including the creation, the implementation, and maintenance of the ontologies. A comparative study of ontology editors is presented in (Kapoor and Sharma, 2010).

For the evaluation of the ontologies, different tools are available, i.e., HermiT (Glimm et al., 2014) and Pellet (Parsia and Sirin, 2004) reasoners. These tools are applied to ensure the consistency of the ontology as well as its overall quality. They can also be used to infer additional knowledge explicitly included in an OWL ontology (e.g., class equivalence checks). That is why they are usually called semantic reasoners or reasoning engines.

OWL ontology is mainly used as a knowledge base for the building of more sophisticated software that is not going to be based on the usage of ontology editors and reasoners. Accordingly, several semantic frameworks have emerged to facilitate the building of semantic-based applications such as OWL API (Horridge and Bechhofer, 2011), OWLReady (Lamy, 2017), Jena, RDF4J, etc. OWL API is mainly used for building and managing OWL ontologies. It provides objects and functions for manipulating the elements that compose an ontology (i.e., classes, individuals, properties, annotations, restrictions, etc.). Jena and RDF4J are the frequently RDF-based frameworks used to extract data from and write to RDF graphs.

1.2.4 Modular ontology

An ontology module may be defined as: “a reusable component of a larger or more complex ontology, which is self-contained but bears a definite relationship to other ontology modules” (Stuckenschmidt et al., 2009). Based on the definition 1.1, we formally define an ontological module as follows:

Definition 1.2.4. An ontological module denoted Mo is a 4-tuple: Mo=<CP_{Mo}, C_{Mo}, R_{Mo}, A_{Mo}> $\in$ O
Where,

(i) $CP_{Mo} \in C_{Mo}$: is the pivotal class of the module,

(ii) $C_{Mo} \subset$ C: represents the set of classes of the module,

(iii) $R_{Mo} \subset C_{Mo}$ x C_{Mo}: is the set of relationships among ontological module classes,

(iv) $A_{Mo} \subset$ A: is the set of axioms that refer to assertions and rules in a logical form.

Ontology modularization or modular ontology is an interesting approach that deals with ontology re-usability. It is the process of defining a module, which is a subset of the main ontology. The modularity of the ontology aims at:

- Improving the ontology development process by reducing the complexity of designing, maintaining, enriching and replacing modules,
- Maintaining the clarity and the coherence of the ontology by presenting ontology modules with needed knowledge, and
- Promoting the reuse of each module separately.

1.3 Data integration

Data integration is the process of combining data retrieved from multiple and independent sources to provide the user with an integrated and unified view of data, called global schema (Lenzerini, 2002). Data integration has been the subject of research since the early eighties (Smith et al., 1981) (Dayal and Hwang, 1984). Since then, theoretical and applied research in this area has been very important.

Data integration grew out of Extract, Transform, and Load (ETL) tools in order to automate efforts. ETL tools extract data from various and heterogeneous sources, then transformed it into a consistent and unified format and finally loaded it into a data warehouse. This process is commonly referred to as the materialized (or warehouse) approach.

However, along with the heterogeneity of data formats and schema, ETL solutions become inappropriate for a flexible and dynamically data management system. New integration approaches are required to handle the massive amount and the variety of the generated data. This limitation has drawn the interest of many researchers to NoSQL data management systems, as these systems should provide data management features for a high amount of schema-less data. Some of them have already gained recognition due to adoption by large projects. Among them: Hadoop/HBase, Cassandra, SimpleDB, MongoDB, and CouchDB.

Nevertheless, NoSQL systems did not solve the heterogeneity problem. They just added new choices for data management (new data models, new query languages, etc.) Therefore, the need for integrated access to all data in organizations is more prominent than before. As an answer to this situation, Data Lake systems have emerged. “A data lake is a set of centralized repositories containing vast amounts of raw data (either structured or unstructured), described by metadata and organized into identifiable data sets” (Chessell et al., 2014). The proliferation of data lakes enabled the switch from ETL to ELT (Extract, Load, and Transform) approach.

However, data is useful only if they can be used to make right and timely decisions. Spending a lot of time in finding data in a data lake reduces the efficiency of data management systems. Accordingly, data lake systems cannot handle the scale and complexity of huge data. Additionally, they do not provide real-time analysis and situational awareness that operators and engineers need to make critical operational decisions at the moment. Experts and software agents need a more flexible approach to integrate data (of the Data Lake, 2019). Therefore, the field of data virtualization began to develop. Data virtualization enabled a more agile

approach to data integration.

1.3.1 Data integration architectures

Integration can either be done physically or virtually. Regardless of the approach category, in general, an integration system is composed of three main layers, as shown in Figure 1.4:

- **Data layer:** It contains all the data sources to be integrated.
- **Warehouse or mediator layer:** It contains the necessary elements to query the different sources through a global schema. This latter can be virtual or materialized in a data warehouse.
- **Wrapper or Charger Layer:** This layer provides access to data sources, extracting data from them, and representing the data in the global schema. It is also the means by which the source can interact with the other components of the architecture.

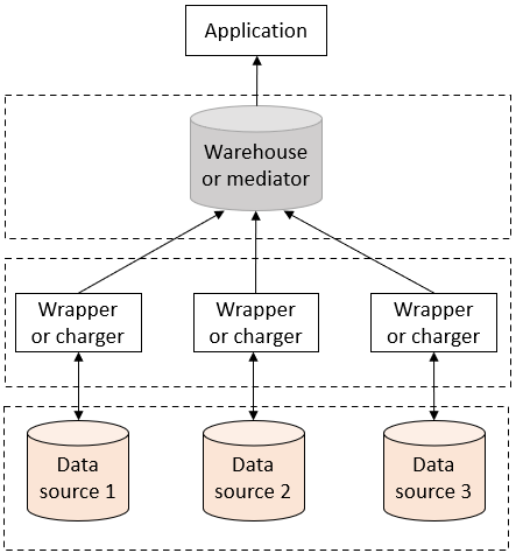


Figure 1.4: Data integration system.

Conventionally, a data integration system is defined as follows:

Definition 1.3.1. A data integration system I = ⟨G, S, M⟩, where:

- G is the global schema used to represent the unified view,
- S is the set of data sources that are represented by the set of local schemas S1, . . . , Sn,
- M represents the mappings that specify correspondences between concepts of the local schemas and concepts of the global schema.

In the materialized approach, the data sources are physically integrated, and the global schema is fully materialized (see Figure 1.5). Indeed, it is necessary that a new base, i.e., a warehouse, be developed via a DBMS with a reproduction of all source data corresponding to the global schema. In this case, requests to the global schema of the warehouse are based on traditional interrogation techniques in the field databases. As a result, the user interacts with the warehouse through direct data query requests.

Data integration is then based on the overall warehouse schema providing an integrated view of sources. Since the integration is performed by ETL tools, which perform a multi-phase process, the data integration process is divided into four main steps (Hacid and Reynaud, 2004) that correspond to those of the ETL (Extract, Transform and Load) approach below.

1. Data extraction from sources.
2. Data transformation at the structural and semantic levels.
3. Data integration.
4. Storage of data integrated into the target system.

The main advantages of materialized approaches are reactivity and the performance of the system. Since the data is gathered into a single source similar to that of a simple database, query processing is centralized, fast, and efficient. Furthermore, system performance is no longer dependent on the performance of each source or communication delays. Nevertheless, materialized approaches are expensive in maintenance and implementation, and ET processing is time-consuming (Chen et al., 2004). Besides, there is no guarantee that the data loaded into the data warehouse is up-to-date. In virtual approaches, the global schema is entirely

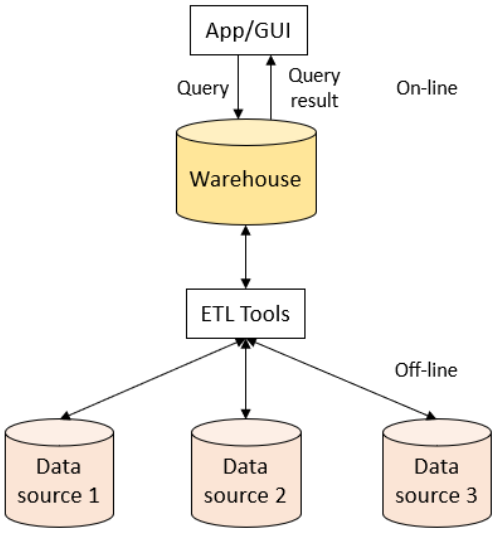


Figure 1.5: Materialized data integration system.

virtual and thus not materialized. In this case, all data remains in local sources and is

accessed through an intermediate infrastructure, usually called a mediator that contains the global schema. The objective of this approach is to give the user the illusion of querying a homogeneous and centralized system, while the sources are distributed and autonomous. In virtual approaches, we distinguish between mediator-wrapper architecture and federated architecture (Hose et al., 2011).

The mediator-wrapper architecture was proposed by Wiederhold to integrate data from a selection of independent sources (Wiederhold, 1992). It is composed of three layers (Figure 1.6): mediator, wrappers, and sources. The mediator provides a common interface to the user that is used to formulate queries. Specifically, At the mediator level, the global schema provides a vocabulary for expressing requests and describing sources through a set of abstract views on them. As a model of the field of application, ontology can be used as a global schema. Indeed, it provides a structured vocabulary to support the expression of requests. Besides, the mediator establishes a connection between the accessible sources by describing their content in a consistent and uniform manner, in a global catalog.

Wrappers are tools that allow mediators to access the content of the sources in a uniform language. They decompose or reformulate (rewrite) a request in a specific query language accepted by each source. Wrappers also take care of transforming the data that the sources produce as answers to the query into the mediator's global schema. The second variant of

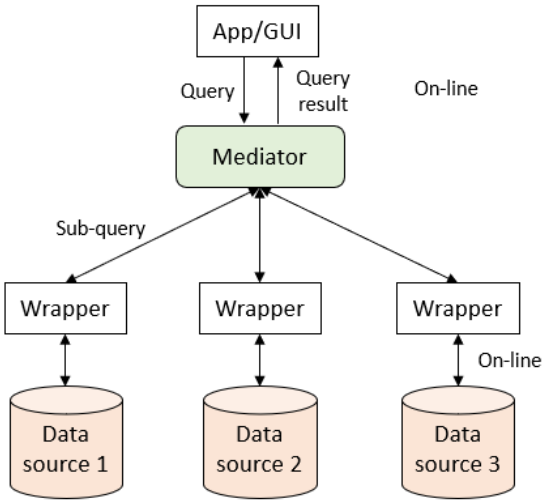


Figure 1.6: Mediator-wrapper-based data integration system.

virtual integration approaches is *federated approaches*. A federation is "a consolidation of multiple sources providing a common interface" and, therefore, very similar to the mediator-wrapper-based approach. The main difference to mediator-wrapper-based approaches is that sources support the global schema and the query language that the federation agreed upon. However, from the user's point-of-view, there is no difference between the two architectures as both allow transparent access to data. Similarly, the Semantic Web community does not distinguish between these two architectures, so both approaches are mainly called federated systems, virtual integration, or federated query processing (Hose et al., 2011).

In this context, different types of SPARQL query federation can be defined that differ on the access mechanism at the sources. They are divided into three main categories: Query federation over multiple SPARQL endpoints, Query federation over Linked Data, and Query federation on top of Distributed Hash Tables (Saleem et al., 2016).

- **Query federation over multiple SPARQL endpoints:** In this type of approach, RDF data is made available via SPARQL endpoints. The federation engine makes use of endpoint URLs to federate sub-queries and collect results back for integration. This type of approach is the most popular used one.
- **Query federation over Linked Data:** This type of approaches relies on the Linked Data principles for query execution. The set of data sources that can contribute to extract results is determined by using URI lookups during the query execution without any data knowledge. Accordingly, this type of approach cannot guarantee to find all results because the relevant data sources change according to the starting point.
- **Query federation on top of Distributed Hash Tables:** This kind of federated approach stores RDF data on top of Distributed Hash Tables (DHTs). However, many of the LOD datasets are not stored on top of DHTs.

Following these explanations, we turn our attention to the SPARQL endpoints federation systems. Figure 1.7 illustrates the architecture of a SPARQL endpoint federation system. This latter is composed of three layers: federator, SPARQL endpoints, and data sources (Rakhmawati et al., 2013). In this type of system, data sources are RDF stores, and they are available via SPARQL endpoints. A SPARQL endpoint is a query processing service based on the SPARQL HTTP protocol, which enables to query remote data sources for both machines and humans. The federator (also called federation engine) makes use of endpoint URLs to federate sub-queries and collect results back for integration. Despite the differences

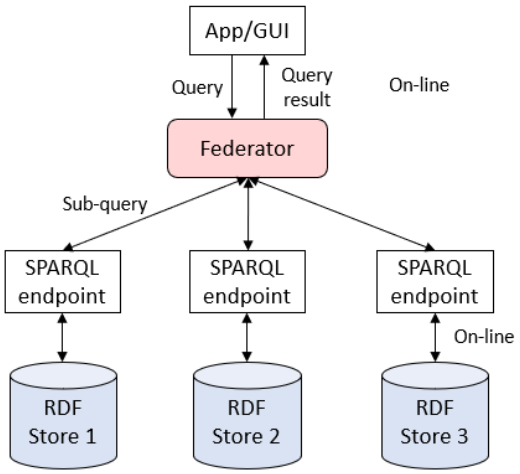


Figure 1.7: SPARQL endpoint federation system.

in architecture, virtual data integration approaches have the advantage of avoiding the cost of materialization. Furthermore, the integrated data is always up-to-date.

1.3.2 Semantic data integration with ontologies

In recent years, the focus has been on data integration, relying on SW technologies. Specifically, an ontology can be used as a global schema. An ontology allows interoperability between various heterogeneous data sources by providing a structured vocabulary that describes their content in a homogeneous and uniform manner.

Some of the best-known semantic data integration approaches include approaches based on the Ontology-based data Access/integration (OBDA/OBDI). OBDI approaches are classified as virtual data integration approaches and refer to the use of ontologies that capture implicit knowledge across heterogeneous relational databases to achieve semantic interoperability between these sources (Wache et al., 2001). In an OBDI, a domain ontology is connected to the data sources through an explicit representation given in terms of mappings that relate symbols in the ontology (classes and properties) to views over the data expressed by means of SQL queries. In the particular case where the organization manages a single data source, the term ontology-based data access (OBDA) system is used. Formally, we define an OBDA specification and an OBDA system, as stated in definition 1.3.2.

Definition 1.3.2. An OBDA specification J is a triple <O, S, M>, where:

- O is an ontology,
- S is the schema of the data source,
- M is the mapping between O and S. Specifically, M consists of a set of mapping assertions, each one relating a query over the source schema to a query over the ontology.

An OBDA system <J,D> is obtained by adding to J an extensional level, which is given in terms of a database D, representing the data at the source, structured according to the schema S (Calvanese et al., 2017).

The ontology and the mappings together expose the data in the form of a virtual RDF graph, which is not materialized. These virtual RDF graphs can be materialized, generating RDF triples stored in RDF triple stores; alternatively, they can be kept virtual and queried using SPARQL queries. These queries are translated, making use of the mappings into queries over the data sources (SQL queries for the relational databases, for example). In this setting, users simply query the ontology and no longer need an understanding of the data sources.

1.3.3 Data integration techniques

1.3.3.1 Schema matching

Schema matching is the problem of finding potential associations between elements (most often attributes) of two schemas. Given two schemas S1 and S2, a solution to the schema matching problem called a schema matching (or more often a matching), is a set of matches. Matchings can be used as input to schema mapping algorithms, which discover the semantic relationship between two schemas (Liu and Özsu, 2009). Specifically, in order to resolve semantic conflicts, it is possible to discover semantic correspondences among the elements of different schemas by correctly identifying the meaning of schema labels (Hossain et al., 2014). This process is called semantic schema matching. Ontologies can be viewed as schemas for knowledge bases. Therefore, techniques developed for semantic schema matching in most of the cases use ontologies as domain knowledge for a semantic annotation (i.e., finding the meanings of a schema label in the ontology).

1.3.3.2 Schema mapping

Schema mapping is the process of establishing semantic connections between schemas. Given a source schema S and a target schema T, a schema mapping M is a specification of a relation between instances of S and instances of T (Liu and Özsu, 2009). Many different mechanisms and algorithms for schema mapping have been proposed. They fall into four categories: *Global As View*, *Local As View*, *Global and Local As View*, and *Both As View*.

a) Global As View (GAV): In GAV, the global schema is described in terms of local schemas (Chawathe et al., 1994). The main advantage of this method is the facility of translating a query on the global schema into queries on local schemas. However, GAV has several drawbacks. First, since the global schema is expressed in terms of the sources, global relations cannot model any information not present in at least one source. Second, GAV-based systems do not facilitate adding a source to the system independently of other sources. Instead, when adding a new source to the system or changing a local source schema, the corresponding mappings must be recreated.

b) Local As View (LAV): To overcome the shortcomings of GAV, researchers came up with the Local As View (LAV) approach (Duschka et al., 2000). LAV follows the opposite direction where each local schema is expressed in terms of the global schema. The main advantage of LAV is the simplicity of adding a new source to the system while preserving the global schema since each data source is described independently of the others. However, LAV suffers from the symmetric drawbacks of GAV. In particular, it cannot model sources that have information not present in the global schema, and each modification in the global schema needs to change all of the source correspondences.

c) Global and Local As View (GLAV): To combine the advantages of both GAV and LAV based solutions, (Friedman et al., 1999) proposed a new category of mapping languages, called Global and Local As View (GLAV). This hybrid solution is a generalization of the GAV and LAV approaches. In GLAV, both assertion types can be used to define the mapping between the source schema and the global schema. GLAV-based systems are also called mediated systems because they usually use an additional mediation layer between the global schema and the data sources.

d) Both As View (BAV): Another hybrid schema mapping approach was proposed by (McBrien and Poulovassilis, 2003) and called Both-As-View (BAV). BAV is based on the use of reversible schema transformations and provides the possibility to derive both GAV and LAV views of the system. The main advantage of this solution is the ability of the evolution of both the global schema and also data sources. Moreover, the benefits of GAV and LAV approaches can be exploited.

e) Mapping languages Several mapping languages have been proposed to represent schema mappings. Here, we focus on semantic-driven mapping languages. D2RQ (Cyganiak et al., 2012) is a mapping language that describes mapping rules between the relational database schema and target ontologies or RDFS vocabularies in order to publish semantic data in RDF format. R2RML (RDB to RDF Mapping Language) is another mapping language, which is a W3C recommendation for expressing customized mappings from relational databases to RDF datasets (Das et al., 2012). RML (RDF mapping Language) (Dimou et al., 2014) extends R2RML by including mappings of various data formats (XML, JSON, CSV) other than relational databases. An RML mapping defines a mapping from any data to RDF. It consists of one or more triples maps. Figure 1.8 presents an overview of RML.

A triples map (*tp*) is composed of exactly one logical source (*property rml:logicalSource*), one subject map (*property rr:subjectMap*) and any number of predicate-object maps (*property rr:predicateObjectMap*). RML logical source extends the R2RML logical table and points to the data source (*property rml:source*); this may be a file on the local file system or data returned from a Web service, for instance. Naming the data source within the mapping makes it possible to map several related data sources simultaneously. A reference formulation (*property rml:referenceFormulation*) names the syntax used to reference data elements within the logical source. As of today, possible values are ql: JSONPath, ql: XPath, ql:CSS3, and rr: SQL2008. Listing 1.6 shows an example of a logical source.

```
<#PrecipitationMapping>
     rml:logicalSource [
  rml:source "precipitation.json";
  rml: referenceFormulation ql:JSONPath;
  rml:iterator "$.[*].Precipitation" ].
```

Listing 1.6: RML logical source.

The subject map (*rr:subjectMap*) includes how to define the subject of each triple and its

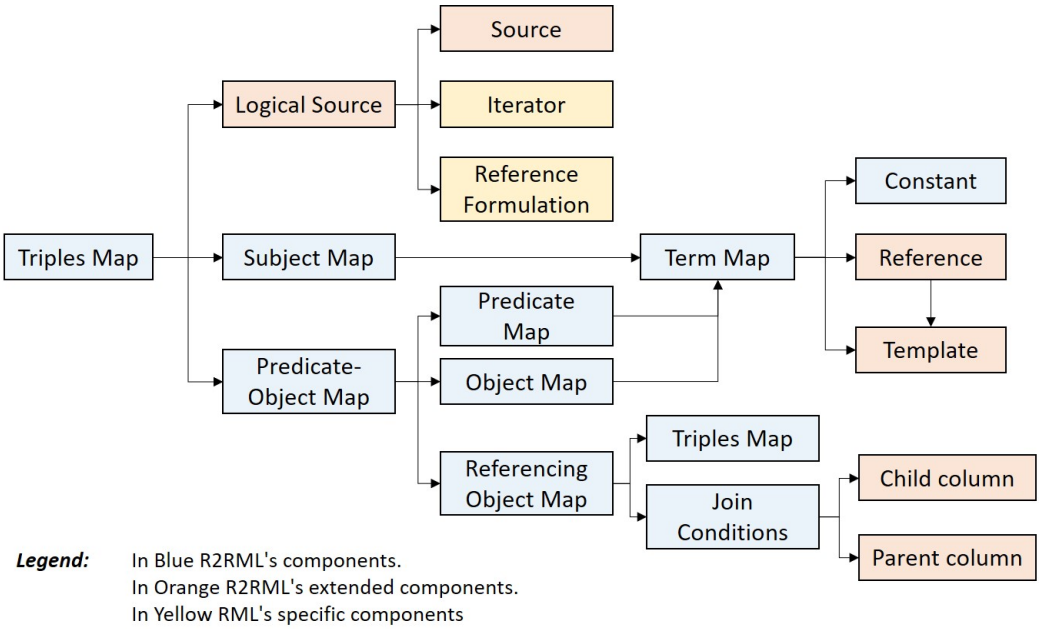


Figure 1.8: RML overview.

optional type of URI. A predicate-object map consists of predicate maps (*property rr:predicateMap*) and object maps (*property rr:objectMap*). Listing 1.7 shows an example of a subject map and a predicate-object map.

```
  <#PrecipitationMapping>
rr:subjectMap [
rr:template "http://example.com/{value}";
rr:class ex:Precipitation ];
rr:predicateObjectMap [
rr:predicate ex:observed_in;
rr:objectMap [ rml:reference "date"] ].
```

Listing 1.7: Subject and Predicate-Object Maps.

The last aspect of R2RML that was extended in RML is the referencing object map. A referencing object map allows using the subjects of another triples map as the objects generated by a predicate-object map. It is represented by the property *rr:parentTriplesMap*. Listing 1.8 shows an example of a predicate-object map with a referencing object map.

```
  <#PrecipitationMapping>
       rr:predicateObjectMap [
    rr:predicate ex:observed_in;
    rr:objectMap [
      rr:parentTriplesMap <#TriplesMap2>] ].
```

Listing 1.8: A predicate-object map example with a referencing object map.

RML mappings definitions are expressed as RDF graphs. These RDF graphs can be kept

virtual and queried online or can be materialized by generating RDF triples. Listing 1.9 illustrates a case of JSON document as input and Listing 1.10 the output in RDF form:

```
(Precipitation.json)
[ ... { "Precipitation" : [{ "Value" : "39 mm", "city": "Paris", "date
   ": "20181230"}, { "Value" : "30 mm", "city": "Paris", "date":
   "20181229"}]...]
```

Listing 1.9: Inputting in JSON format.

```
(Output RDF representation)
"http://example.com/39 mm" a ex:Precipitation;
ex:observed_in "20181230".
"http://example.com/39 mm" a ex:Precipitation;
ex:observed_at "Paris".
"http://example.com/30 mm" a ex:Precipitation;
ex:observed_in "20181229".
"http://example.com/30 mm" a ex:Precipitation;
ex:observed_at "Paris".
```

Listing 1.10: Outputting in RDF format.

1.4 Query processing

Once a data integration system is established, end-users can express queries over the global schema. However, query processing (also called query answering) differs according to the integration approach category. In the materialized approach, a copy of all data which corresponds to the global schema is saved in a global and single database. Thus, a query against the global schema is answered by simply running the query over the materialized global database. On the other hand, in virtual data integration, data are kept in the sources, and thus, a query against the global schema has to be translated to corresponding queries against the local schemas (Katsis and Papakonstantinou, 2009). Here, the query processing makes use of mappings by decomposition and reformulation of the user query into sub-queries over information sources and then re-composition of partial answers for a unified response for the original query. Accordingly, the query processing method depends on the mapping approach that the virtual data integration system chooses.

1.4.1 Query processing types

The query processing is undertaken in different ways depending on the schema mapping category. In the GAV approach, the global schema is expressed in terms of the data sources and each global relation is defined as a view over the local relations. However, since the local schemas are autonomous, it may happen that tuples in a global relation do not exist in local relations or that a tuple in a global relation appears in different local relations. Accordingly, this lack of completeness may yield incomplete answers to queries.

Unlike the GAV approach, the query processing in LAV offers query optimization, which enables the system to select a minimal number of data sources relevant to a query. However, finding the correspondences between the terms used in the global schema, and those used in the local schemas requires comparison with each local schema, thus making schema comparison time-consuming.

The hybrid approaches (GLAV, BAV) express both a global schema and local schemas by using LAV and GAV query rewriting processes to reduce the LAV complexity while ensuring the global schema scalability.

1.4.2 Distributed query processing principles

Given the fact that data is distributed over various sources, a distributed query processing strategy will decompose the original query into subqueries that are relevant for each data source and later join local results. Distributed querying typically implies five main steps (Özsu and Valduriez, 1999): query parsing, source selection, query rewriting, query planning, and query execution, as depicted in Figure 1.9.

- *Query parsing* is the first task performed in any query processing system. This task aims to ensure that the query is correctly specified—that is, it's well-formed according to the query language syntax—and to convert the query into an internal pattern in order to facilitate the manipulation of the query for later steps, particularly, query rewriting and query planning. For instance, a SPARQL query could be transformed into a Basic Graph Pattern (BGP) or an abstract syntax tree.

- *Source selection* (also called *data localization*) identifies the relevant data sources for each triple pattern or set of triple patterns of a query. In fact, all data sources are not necessarily containing data that are relevant for a given query. Thus, selecting relevant sources prevents from sending useless queries. In the literature, we have identified two main source selection methods: Ask query and Data catalog. ASK query returns a boolean value that decides whether the query can be answered by the data source or not. In the second method, by looking up data catalogs of sources, the mediator can predict the relevant sources for an input query.

- The *Query rewriting* step decomposes the original query into a set of sub-queries that will be distributed to the different data sources. It both ensures the preservation of the original query semantics and takes into account the performance of the sub-queries to be processed.

- The *Query planning* step sorts sub-queries in order to generate an optimal evaluation plan that minimizes the overall query execution time. As a general rule, the most selective queries should be processed first.

- Finally, the *query execution* (or *query evaluation*) step is intended for the execution of the query plan and the collection of the results generated from all the sub-queries. To

this end, several strategies can be executed, including nested loop join, bind join and hash join (Rakhmawati et al., 2013).

An important aspect of query processing is query optimization. The goal of query optimization is to choose the best execution strategy for a given query under the given resource constraints (Özsu and Valduriez, 1999). Hence, query optimization was developed as a significant area of research for distributed query processing, and many related works primarily address the problem of query performance. The optimization can be performed in the different query processing steps and is categorized as static and dynamic. Static optimization refers to optimization processed before the query execution (source selection, query planning, and query rewriting steps). Here, the optimization techniques aim at improving the accuracy and completeness of the query response. On the other hand, dynamic optimization refers to the optimization techniques done during the query execution phase in order to reduce the processing time and the communication cost between the query engine and remote data sources.

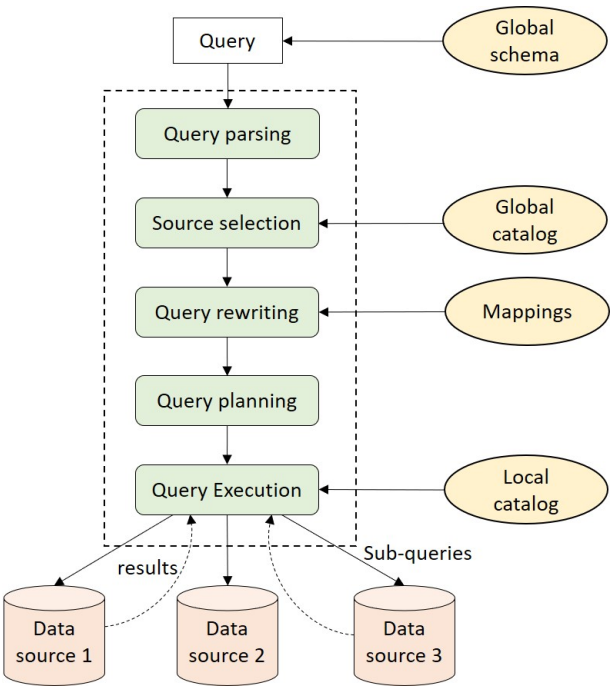


Figure 1.9: Distributed Query Processing steps.

Query optimization is said to be semantic if the transformation of the query relies on semantic knowledge (Kermanshahani, 2009). Semantic query optimization is one of the most efficient ways to perform query optimization in the context of data integration.

1.5 Hypergraphs

Hypergraphs are able to describe complex systems as their descriptive power is fairly strong because they are one of the most general graphs and mathematical structures for representing relationships (Molnár, 2014).

A hypergraph is the generalization of an ordinary graph by introducing hyperedges, which are non-empty subsets of the vertex set. Vertices of a hypergraph represent the entities to be modeled, such as concepts. Hyperedges represent the high order relations between those entities. In what follows, we provide a set of definitions presented by (Bretto, 2013).

Definition 1.5.1. Hypergraph A hypergraph H is a pair < V, E>, where

(i) V=v_1,...,v_n is the set of vertices or nodes,

(ii) E = $(e_i)_{i\in I}$, (I is a finite set of indexes) is the set of non-empty subsets of V, called hyperedges where each $e_i \in$ E is a subset of V.

A definition of a sub-hypergraph can be given based on the hypergraph definition.

Definition 1.5.2. Sub-hypergraph A sub-hypergraph H(V') of the hypergraph H is the pair <V', E'> where

(i) V'$\subset$ V,

(ii) E' = $(e_j)_{j\in J}$ such that for all $e_j \in$ E': ej $\subseteq$ V' and J $\subseteq$ I

The notion of hypergraphs may be generalized in a way that the hyperedges can be represented in certain cases as vertices.

Definition 1.5.3. A Generalized hypergraph GH= <V, E> is an hypergraph where a hyperedge $e_i \in$ E may consist of both vertices and hyperedges as well. The hyperedges that are contained with the hyperedge e_i should be different from e_i.

Definition 1.5.4. A directed hyperedge (hyperarc) $\vec{e_i}$ is an ordered pair :

$$\vec{e_i}=(\vec{e_i}^+=(e_i^+,\text{i});\vec{e_i}^-=(\text{i},e_i^-)),$$

where $e_i^+ \subseteq$ V is the set of vertices of $\vec{e_i}^+$ and $e_i^- \subseteq$ V is the set of vertices of $\vec{e_i}^-$. The elements of $\vec{e_i}^+$ (hyperedges and/or vertices) are called the tail of $\vec{e_i}$ while elements of $\vec{e_i}^-$ are called the head of $\vec{e_i}$.

Figure 1.10 illustrates an example of generalized hypergraph where:
V = {v_1; v_2; v_3; v_4; v_5}
E = {e_1 = {v_1, v_2}; e_2 = {v_3, e_1}, e_3= {(e_1, v_5), v_4}}.
$\vec{e_3}$ is a directed hyperedge consists of a tail e_1, v_5 and a head v_4.

Hypergraphs have attracted increasing attention of researchers. They were applied in several domains and applications such as social network systems, service-oriented applications,

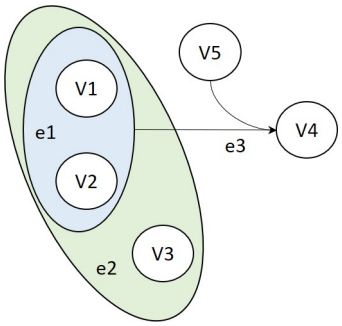


Figure 1.10: A generalized hypergraph example.

information systems, and even data integration (Theodoratos, 2002),(Saleem and Ngomo, 2014), and they have proved their efficiency. From our perspective, we think it might be interesting to take advantage of hypergraphs' benefits and use them in the semantic data integration context.

1.6 Conclusion

According to a world economic forum report, around 70% of available data are not used due to the lack of interoperability and linking of data that are in isolated silos (Report, 2019). Undeniably, the integration of heterogeneous different data sources is a crucial necessity, and this is an area, in particular, where semantic technologies shine. However, various challenges exist to integrate, manage, and exploit this multi-source data in a semantically homogeneous way.

In this chapter, we have described the fundamental challenges related to the semantic data integration context. We also have discussed schema matching, schema mapping, and query processing, which are three fundamental tasks when developing a data integration system. In the following chapter, we will introduce the context of the PREDICAT project, scope of this thesis, and we will provide a detailed state of the art of data integration approaches and existing ontologies related to the environmental monitoring domain.

Chapter 2

Context and State of the art

Sommaire

2.1 Introduction 31
2.2 Scope of this research 32
2.2.1 Applicative context 32
2.2.2 Data challenges 33
2.2.3 Scope: PREDICAT project 35
2.2.4 Focus of this thesis 38
2.3 State of the art of ontologies related to EO 39
2.3.1 Sensor ontologies 39
2.3.2 Observation and measurement ontologies 40
2.3.3 Environmental monitoring ontologies 41
2.3.4 Synthesis 45
2.4 State of the art of semantic data integration approaches 45
2.4.1 Materialized approaches 45
2.4.2 Virtual approaches 47
2.4.3 Synthesis 50
2.5 Discussion and orientations 55
2.6 Conclusion 57

2.1 Introduction

Driven by the need for natural disasters prevention, environmental monitoring has been the subject of research and ongoing scientific investigation of several governments/organizations such as the case of the Franco-Tunisian PREDICAT project, in which this thesis is elaborated. In this chapter, we will start by defining the context and the challenges of the project, followed by a brief description of the PREDICAT platform, as well as the focused challenges of our research work (2.2. According to these challenges, we will drive an extensive literature review of existing works on ontologies related to the EO domain and those on semantic data integration and query processing approaches. Initially, section 2.3 provides an analysis of sensor ontologies, observation and measurement ontologies, and environmental monitoring

ontologies. section 2.4 discusses existing materialized and virtual semantic data integration approaches and gives an overview of existing SPARQL query federation systems with respect to source selection type since this latter represents a fundamental task when developing a data integration system. Based on the identified research gaps, we underline our orientation in section 2.5.

2.2 Scope of this research

In this section, we present an overview of the thesis context from a scientific point of view. This thesis is part of the Franco-Tunisian project PREDICAT. We will start by defining the context and the motivations of the project, followed by an understanding of the project challenges. Next, we will present the PREDICAT platform architecture and its main components. At the end of this section, the focused challenges and the orientations of the thesis are presented.

2.2.1 Applicative context

Natural disasters such as hurricanes, floods, and earthquakes are the result of occurring weather anomalies and hazardous events. Over the past 30 years, there has been a substantial increase in all types of disaster situations. These latter have affected millions of people, and thousands of people have died because of disasters every year (Saulnier et al., 2019). The damages they have caused to human society were horrific. Driven by the need to cope with disaster situations, several EO programs and organizations, including, the Copernicus program, NOAA, and OSS, have emerged for sharing and monitoring EO data. EO programs are used to observe, monitor, and assess the status of and changes in the environment. These systems are becoming increasingly important so that a vast amount of EO data is daily collected.

Indeed, the success of a disaster forecasting process is mostly dependent on better exploitation of EO data to understand environmental phenomena and analyze the influencing factors or disaster chain information. However, despite the availability of large amounts of data, the usage of EO data is still limited due to the lack of interoperability and data linking (Athanasiadis, 2015). Several facts illustrated that the interoperability difficulties among EO systems are a significant challenge. The Hurricane Irma, which occurred across the Caribbean in 2017, serves as an example. In fact, the traditional conditions of hurricane development (such as the sea level, the wind speed, and the atmospheric pressure) were monitored, as usual, by the NOAA National Hurricane Center. Besides, the African Sahara Desert was observed by NASA. However, there was no link between these observations. Indeed, the absence of the dry air resulting from the lack of Saharan dust across the Atlantic acted in favor of high-altitude winds (IRMA, 2017). When these waves of air have enough moisture, lift, and instability, they readily form clusters of thunderstorms. Thus, a tropical cyclone was formed as the areas of disturbed weather moving westward across the Atlantic, resulting in the creation of the hurricane Irma. If experts had the link beforehand and understood the

phenomena better, they might have been able to predict the power of the disaster a little bit before and alert the governments to set up more preventive actions.

The floods occurred in the Aude in October 2018 presents another example that shows the lack of an interoperable global view of environmental data (French-news, 2018). Indeed, severe flooding caused devastation in parts of south-western France. Consequently, several people develop a very rapid rise in water, especially in Trèbes, where the water reached 7.68m on the morning of Monday, October 15, causing the most violent flood since 1891 and left at least 11 people dead. Soon, voices were cast to terminate a lack of prevention . "I think the system does not respond to the problem because it is fragmented. We have meteorologists on one side looking at the atmospheric sections, and hydrologists on the other side looking at the river sections. We must have a multidisciplinary view of risk, why not create a global center," suggests Emma Haziza, a hydrologist and president of the Mayane research center on flood risk management .

Accordingly, integrating this massive amount of heterogeneous data to provide what we call a global information view, where different systems and programs will have unhampered and uniform access to the available EO data, is becoming crucial. A common information space regarding the surrounding environment allows the data sources to speak the same language and to link and share information so that domain experts and software agents could transform them into actionable knowledge. However, a global information view is further challenged by semantic data integration. Data integration is the process of combining data retrieved from multiple and independent sources to provide an integrated and interoperable structure (Lenzerini, 2002). Semantic data integration is the process of combining and consolidating disparate data into meaningful and valuable information by creating links between them in order to create a richer global view (Cheatham and Pesquita, 2017). The main issue related to semantic data integration is the growth and the diversity of data in terms of semantics, types, and formats, each one compliant to a different standard.

2.2.2 Data challenges

To address the problem of integrating independent EO systems and programs, several data challenges arise, such as data large-scale and data heterogeneity. The term Big Data encompasses all these aspects. It refers to different volumes of data coming from multiple sources. Doug Laney articulated the mainstream definition of big data as the three V's: Volume, Variety, and Velocity (Laney, 2001). Recently, three more V's, that data scientist must be concerned with, were added; Variability, Veracity, and Value of the data. These V's are explained as follows:

- The volume (1st V) describes the massive amount of data that experts are currently dealing with. It has required scientists to rethink storage and processing paradigms in order to develop the tools needed to analyze it.
- The variety (2nd V) refers to the heterogeneity of data, which concerns differences in the data model, the data manipulation language, the data manipulation mechanism,

competition control, etc. Amit Sheth distinguishes between three types of data heterogeneity (Sheth, 2015):

  - Syntactic heterogeneity is caused by differences between software platforms and the type of syntax (for example, XML or spreadsheet syntax) they handle. Indeed, data comes in all kinds of formats from structured data in traditional databases, semi-structured data in CSV, XML, JSON formats to unstructured data such as text, documents, video, and images.
  - Structural (or schematic) heterogeneity occurs when equivalent concepts are represented differently in data sources. It is associated with the choice of names, data types, attributes, or units to build the source diagram. In fact, each data source adopts its own data model or schema that differs from one source to another.
  - Semantic heterogeneity denotes the differences in modeling the same domain of interest (e.g., Synonymy, polysemy, etc.). The problem of having gaps in understanding concerning the same information sometimes comes from the diversity of the geographical and organizational aspects that use them. For example, in the sentence "Maps of daily temperature and precipitation are produced", an expert would recognize that the observation is "temperature" but would not be able to determine the details related to the temperature concept (atmospheric temperature, sea surface temperature, etc.). He needs to ask the data provider to get more details. Semantic heterogeneity is mainly caused due to the use of different terms for defining concepts or due to the use of totally different concepts. We can distinguish a difference between the conceptualization mismatch and the explication mismatch. In the former, various terms may correspond to the same meaning. For instance, OSS uses the word "rainfall" for the same real-world feature that usually refers to "precipitation" in other sources. The variety of terms complicates the work of emergency responders who should be familiar with the terms used in each discipline. On the other hand, in different disciplines, the same term may correspond to various meanings (the explication mismatch). For example, the term "environment" is defined as "the biological and abiotic elements surrounding an individual organism" in the biological domain. However, it refers to "all the natural components of the Earth (air, water, oils, etc.)" in the EO domain (Sauvé et al., 2016).

- The velocity (3rd V) represents data streams at a random speed and must be dealt with in a timely manner.
- The veracity (4th V) deals with the uncertainty of data as they can change over time, impacting on the quality and reliability of data.
- The evolving nature and variation of the data and how the system copes with such changes refer to the variability (5th V).
- Finally, the value (6th V) represents a characteristic describing the main objective of collecting such a huge amount of data, i.e., finding relationships that are explicitly or hidden within the data in order to value it.

Two other significant issues that EO experts and software agents can face while integrating data: unforeseen costs and unauthorized access to data. Indeed, the problem of managing data derived from EO programs, in terms of access, pricing, and data rights, is commonly tricky. On the one hand, APIs and data access services are, in most cases, unavailable. Experts should download each data source with which to work. On the other hand, data may be extracted only after agreeing to specific laws and regulations. Access to data requires to be user-friendly to reach common understanding and decision making for various prediction systems.

Currently, there are several ongoing projects such as EOPEN and beAWARE, that are aiming to solve the integration problem (Gialampoukidis et al., 2017) (Karakostas et al., 2018). These projects similarly aim to semantically integrate heterogeneous data coming from Big data sources such as sentinel data, including data provided by citizens through social media. However, data storage and management with traditional data management platforms will be difficult (Siddiqa et al., 2017). As the number of data sources and the type of data stores augment, data access needs to be made easier for better real-time prediction.

2.2.3 Scope: PREDICAT project

In order to deal with the aforementioned challenges, the Franco-Tunisian project PREDICAT (PREDIct natural CATastrophes) is launched in 2017. This project aims at providing a semantic service-oriented platform for data interoperability and linking in EO and disaster prediction. This three-year project is the outcome of a collaboration between the National School of Engineering of Tarbes (ENIT), the National School of Computer Science of Manouba (ENSI), the University of Lyon 3, and the Higher Institute of Informatics and Multimedia of Sfax (ISIMS). It is funded by the "PHC Utique" program of the French Ministry of Foreign Affairs, managed by Campus France, and the Tunisian Ministry of higher education and scientific research, led by the CMCU (project number 17G1122/ CODE CF 37T03 NJ).

The vision of this project is to ensure a uniform service-based access and semantic data integration of heterogeneous and multi-source EO data, aiming at user-friendly decision-making during extreme weather crisis and disaster prediction, through a platform that is designed to provide a flexible solution. In particular, the focus is on:

1. Formalizing the knowledge of the EO domain to ensure semantic interoperability between the multi-source EO data.
2. Offering to the EO experts a global view of data through semantic linking and integrating of EO data coming from several sources such as NOAA and OSS.
3. Providing a decision support solution to analyze in real-time all the useful data in order to effectively prevent and/or react against natural disasters, through semantic linking of information.
4. Providing reasoning mechanisms.

5. Providing adequate services to access and extract data with any format or structure, in real-time, in order to guarantee faster data management.

6. Producing warnings and real-time decisions to effectively prevent natural disasters.

7. Offering a user-friendly interface allowing to dialog with the PREDICAT platform.

Figure 2.1 presents the global architecture of the PREDICAT platform and its layers (Masmoudi et al., 2018). The layered architecture is composed of seven tiers, namely: (1) data layer, (2) service layer, (3) data processing layer, (4) semantic layer, (5) data integration layer, (6) decision support layer and (7) users' interface layer. The different layers are described in the sections below.

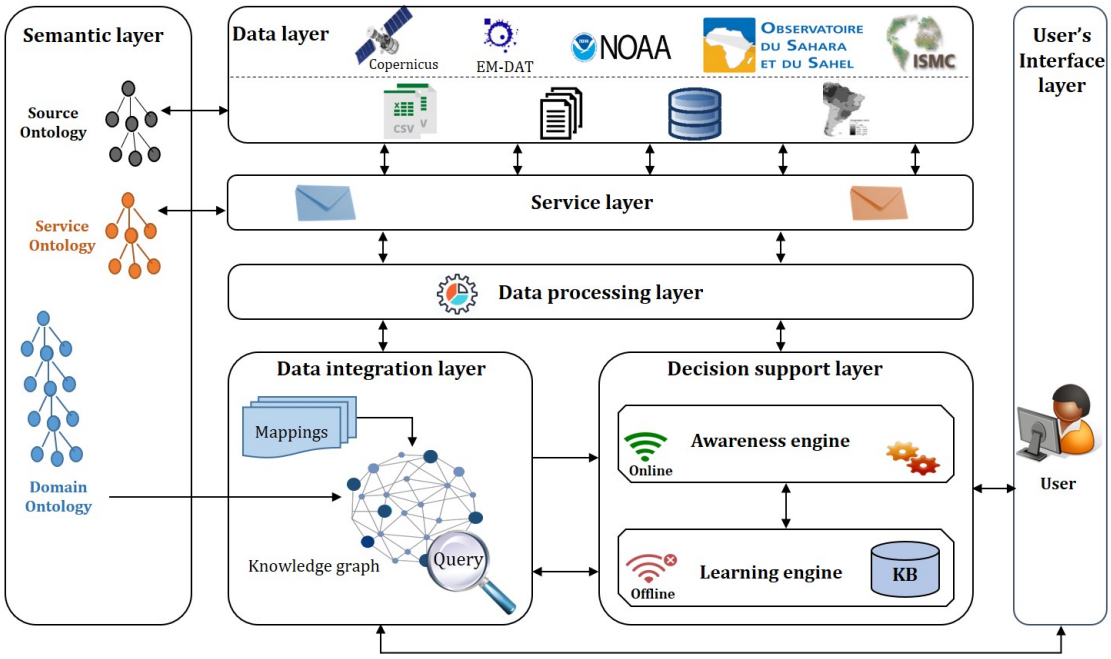


Figure 2.1: PREDICAT architecture.

Data layer: This layer encompasses different data sources relevant to EO and deals with different data format types (i.e., CSV, RDB, ENVI, etc.). For instance, OSS uses satellite imagery to create gridded rainfall time series, and data are downloadable via FTP links[1]. Most of the data sources defined in this layer are pointing to the heterogeneity related to their software applications and the used storage systems. Since data sources present various data types at an unprecedented rate, they are stored in different dedicated storage systems. Besides, the challenges of this layer relate to the access and the extraction of accurate EO data among the numerous datasets, in real-time, knowing the high update frequency changes.

[1]http://chg.geog.ucsb.edu/data/chirps/

Service layer: This layer deals with the implementation of services, accessing EO data. These services will be annotated with semantic descriptions about the services' contents and their related data in order to promote the interoperability between heterogeneous services.

Semantic layer: In this layer, a suite of ontologies is proposed, i.e., domain, source, and service ontologies.

- Domain ontology: is the vocabulary that will provide a formal representation of the environmental monitoring domain. Such a shared vocabulary will resolve terminology inconsistencies and establish semantic interoperability among data sources. Moreover, it will support the integration and linking of data together in order to build a global interactive network that permits to better understand environmental dynamics and natural phenomena.
- Source ontology: The vision of PREDICAT is to combine data generated from multiple EO systems and monitoring programs. For reasons of data quality control and reliability, it is necessary to keep track of the provenance of the data. Accordingly, a source ontology will be proposed to represent additional information about data. The source ontology will identify each source, describe its name, its features, and the products it provides. It will also include information about authenticity or credibility measures and its relation to other data sources, either they shared common products or not.
- Service ontology: The main purpose of a service ontology is to enable the semantic representational knowledge inherent to services and their related relationships. The service ontology will also consider the interoperability issue between data access services. Typically, services are not compatible with each other, which makes interoperability as a significant issue for a successful data access service composition.

Data processing layer: The data processing layer deals with the process of the execution of services. It tackles several issues, such as reducing time-consuming processes, optimizing time-responses to fasten predictions, and reducing costly-consuming bandwidth for requests. Based on the service ontology, the system will propose an orchestration schema for services accessing data.

Data integration layer: The data integration layer consists of combining a large amount of data generated from heterogeneous sources into a single consistent and global view of data. One of the main problems of data integration is data heterogeneity, discussed in section 2.2.2. This heterogeneity is related to the schema, the format, and the vocabulary used to describe the data. In general, each source has its specific characteristics. The integration is mainly performed at the semantic level, to provide reasoning mechanisms and interoperable solutions, through the semantic linking of information. Indeed, this layer uses the proposed domain ontology related to the environmental monitoring domain to exploit data as an interoperable global knowledge graph in order to have more in-depth analyzes of environmental phenomena.

This layer will also support learning from existing data to enhance decision-making and predict future events. Indeed, for machine learning to be effective, data from the widest possible variety of sources must be used; and this is why data integration plays a key role.

Decision support layer: The decision support layer consists of two sub-layers, i.e., learning and prediction. The goal of the learning sub-layer is to execute predictive models that learn from existing data to predict future trends, outcomes, and behaviors. It takes the knowledge graph from the integration layer as input and then generates new relations that help to deduce knowledge and improve the performance of awareness. The prediction sub-layer handles the real-time data and takes into consideration the inferred knowledge from the previous sub-layer to provide early warnings and decision support.

Users' interface layer: This layer consists of a front-end interface allowing to dialog with the PREDICAT platform. Users may have the possibility to query EO data through this interface, and the queried data sources will be displaying their related resulted data to end-users also through this interface. This latter also enables displaying warnings by the decision support layer.

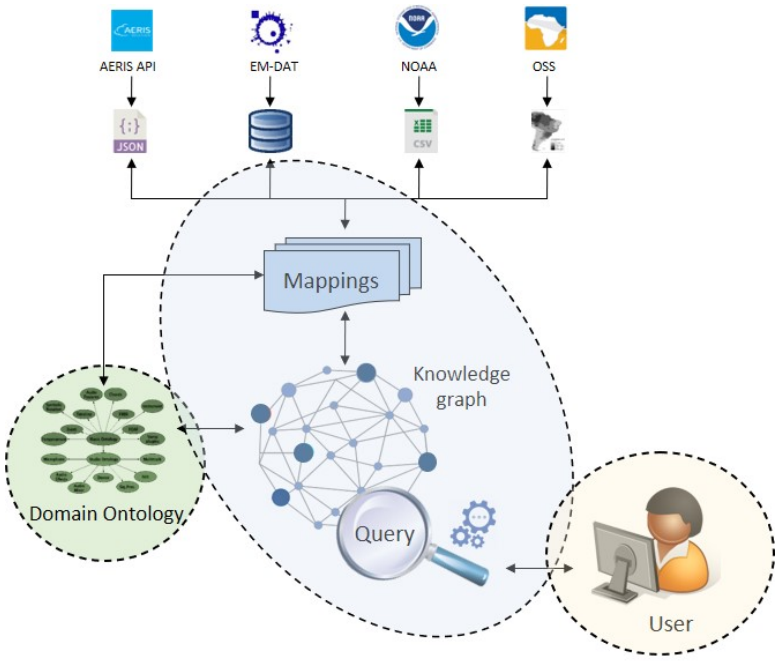


Figure 2.2: Focus of this thesis regarding PREDICAT.

2.2.4 Focus of this thesis

As mentioned before, this thesis is part of PREDICAT project. Specifically, the focus of our work is illustrated in Figure 2.2 and consists in:

1. The semantic layer, precisely, the domain ontology related to the environmental monitoring domain, proposed in order to ensure semantic interoperability among EO data.

2. The data integration layer with the purpose of semantically integrating EO data and creating a common environmental information space.

3. The users' interface layer to query and extract EO data.

Among the challenges defined in section 2.2.2, we will concentrate on the data heterogeneity challenges. Nowadays, there are several approaches and solutions to solve the data volume issue, described by Big Data technologies such as Hadoop and NoSQL databases. Nonetheless, ensuring interoperability among data and understanding it is still challenging and a topical issue.

2.3 State of the art of ontologies related to EO

In this section, we aim at identifying ontologies that can be used to semantically model sensors (Section 2.3.1) and the information available from such systems as a result of the observation of the ambient environment (Section 2.3.2). Next, a review of some existing environmental monitoring ontologies is presented in section 2.3.3. The review is based on a comparison according to the aspects of the knowledge acquisition method, the domain, the principle of reusing existing ontologies, and the building purpose. The current limitations of these ontologies are discussed in section 2.3.4.

2.3.1 Sensor ontologies

The first initiatives for modeling sensor and sensor networks were driven by the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) (Botts and Robin, 2007). The OGC provides a set of XML schemas and open standards that enable the discovery, access, and processing of sensor observations. The OGC SensorML is one of these schemas. It describes sensors, systems, and processes. It also provides the information needed for the discovery of sensors and the location of sensor observations. SensorML was used to support the establishment of the OGC's Sensor Observation Service (SOS), providing access to observations from sensors and sensor systems.

The widely used sensor ontology is the W3C Semantic Sensor Network (SSN) Ontology (Compton et al., 2012). The SSN ontology has been recently updated by including a lightweight core module called SOSA (Sensor, Observation, Sampler, and Actuator) (Haller et al., 2018). This new version of SSN is a joint W3C and OGC standard specifying the semantics of sensors, observations, observable properties, actuation, and sampling. It describes sensors regarding measurement processes, observations, and deployments. However, it does not include concepts about the geospatial, either temporal dimension.

2.3.2 Observation and measurement ontologies

Research on observation and measurement context was driven by different organisms. The Observations & Measurements (O&M) standard of the OGC-SWE, which describes a conceptual model and XML encoding for measurements and observations, is an example. Aligned to the SensorML, the O&M establishes a high-level framework for representing observations, measurements, procedures, and metadata of sensor systems. This standard is required by the SOS, for the implementation of the SWE-enabled architectures.

The SemSOS O&M-OWL ontology is a semantic data model to manage sensor data based on the O&M standard (Henson et al., 2009). The significant concepts modeled in the SemSOS ontology are:

- *Observation:* An act of observing a property or phenomenon, to produce an estimate of the value of the property.
- *Feature:* An abstraction of a real-world phenomenon
- *Property:* associated with a feature that can be sensed or measured.
- *Process:* the method, system, or algorithm used to generate the result (such as a sensor).
- Result data: an estimate of the value of some property.
- *Location:* the location of an observation event.
- *Time:* The time when the phenomenon was measured in the real world.

The significant properties include *feature of interest*, *observed property*, *sampling time*, *observation location*, *result*, and *procedure*.

The Extensible Observation Ontology (OBOE) is a generic ontology dealing with observations data and measurement (Madin et al., 2007). While it's a generic, the concepts of this ontology should be extended to clarify the inherent meaning of scientific observations. The core classes of the OBOE ontology include Observation, Measurement, Entity, Characteristic, and Measurement Standard (e.g., physical units), and six properties labeled hasContext, ofEntity, hasMeasurement, hasValue, hasPrecision, and usesStandard.

NASA's Semantic Web for Earth and Environment Terminology (SWEET) is a mid-level ontology for Earth system science (Raskin and Pan, 2005). It consists of nine upper-level concepts (Representation, Realm, Phenomena, Process, Human Activities, Matter, Property, State, and Relation) that can be used as a foundation for domain-specific ontologies that extend these upper-level SWEET components.

2.3.3 Environmental monitoring ontologies

Several works have attempted to build domain and application ontologies for environmental monitoring. Law-and-Environment Ontology (LEO) is proposed by InforMEA to provide a semantic standard for data, information, and knowledge in the field of environmental law and governance (InforMEA, 2010). It is more a taxonomy since it gives an overview of relationships between Multilateral Environmental Agreements (MEAs), as well as concepts, definitions, and synonyms found in these conventions. However, it is not written in any ontology format. It only contains the information displayed as maps, info-graphics, and text. LEO terms are divided into different sections, such as air, climate, land, and water sections.

Zhang et al. proposed a Meteorological Disaster Ontology (MDO) to describe the components and relationships between the different parts of the meteorological disaster system (Zhang et al., 2016). MDO describes only meteorological disaster knowledge. It does not include other disaster types (like hydrological and geophysical disasters). Also, MDO is not published in a computable format.

Based on the Water Data Transfer Format (WDTF) schema, Shu et al. defined a WDTF ontology using concepts and roles that describe water observation data (Shu et al., 2016). WDTF documents were translated into ontology instances to build an application ontology that presents a semantic solution for WDTF data validation. In that work, Shu focused only on data encoded in WDTF.

Qui et al. proposed an ontology-based approach that links environmental models with disaster-related data to support flood disaster management (Qiu et al., 2017). They defined a model ontology and a data ontology by considering disaster-related semantics and described the relationships of models and data with a multi-level semantic mapping method.

Oliva-Felipe et al. proposed the Waste Water ontology WaWo+ (Oliva-Felipe et al., 2017), an evolution of the original WaWo Ontology. The WaWO ontology was an attempt to build a model defining the meaning of each term/concept used in the wastewater domain. In addition to WaWo vocabulary, the WaWO+ ontology covers the urban water cycle, social, and water quality aspects. It aims to allow the inference of new knowledge from existing facts, allowing agents to have more accurate information about their environment.

However, none of the ontologies mentioned above use a common upper-level ontology nor reuse other domain or existing ontologies. They have been developed from scratch, each in its ad hoc way in such wise that they create a lack of interoperability with other ontologies in related domains. In contrast, the reuse of other domain ontologies was the subject of other works.

Examples include Boughannam et al., who discussed the advantages of semantic technologies in implementing smarter industry solutions for monitoring analytics (Bou-Ghannam, 2013). They proposed an ontology for managing observations and measurements by reusing various existing ontologies and standards, such as the Quantity-Unit-Dimension-Type (QUDT) ontology (Hodgson et al., 2014), SSN, and the OGC GeoSPARQL standard (OGCI, 2010).

The adopted solution illustrates how the concepts of these ontologies are implemented using a real-world use case from the environment analytics domain. The environmental analytics domain ontology proposed by the authors is dedicated to monitoring the environment around their deployed platforms and assigned to a specific use case, which is the oil and gas monitoring. Nevertheless, the ontology does not use an upper-level ontology nor a modularization approach.

Dahleh and Fox created an ontology for the representation of environmental indicators (such as ozone concentration and noise pollution) (Dahleh and Fox, 2016). They built three domain ontologies to represent pollution, sensor, and species information, based on the Global City Indicators (GCI) Foundation ontology (Fox, 2013) and other reused ontologies such as the SSN ontology. This global ontology is specific to environmental indicators representation, and it does not represent additional monitoring information such as events and environmental processes.

However, these works (Bou-Ghannam, 2013) and (Dahleh and Fox, 2016) do not use upper-level ontology. This concern was adopted by other works in the domain of environmental monitoring. SemsorGrid4Env is a project which uses ontology as support for sensor data integration (Gray et al., 2009). Two ontologies based on real use cases were built in this project. The first one is a fire risk monitoring ontology, and the second one is a coastal and estuarine flood ontology. The two ontologies are mainly developed, reusing SSN to represent sensor concepts and their observed information, and ontologies from the SWEET suite to describe services and datasets. Geographic and administrative region data are covered by the Ordnance Survey ontologies.

In the MONITOR project (Kollarits et al., 2009), Kollartis et al. defined an ontology that formalizes the knowledge necessary for monitoring methods and environmental risk management. MONITOR ontology describes the relations between social, natural, hazardous events, and several risks, risk assessment, and risk management terms. It is based on the upper-level ontology DOLCE. However, the ontology is not available in coded form, and its concepts mainly intended for risk management systems.

Buttigieg et al., proposed the ENVironment Ontology (ENVO) which is aligned to BFO (Buttigieg et al., 2016). It delineates the environment domain as a whole by including environmental features, environmental materials, and environmental processes. ENVO also includes other fields such as biomedicine, ecology, food, habitats, and socioeconomic development. Unfortunately, it lacks the ability to implement various concepts and relations for sensor, observation and measurement contexts.

Many studies have applied an ontology-based approach to their emergency management applications. Llaves and Kuhn developed an Event ABStraction (EABS) ontology to model events inferred from observations and an application ontology named flood monitoring ontology (Llaves and Kuhn, 2014). EABS ontology is kept generic enough to be reused for other applications. It is aligned to DOLCE, extends the SSN ontology, and allows only inferring information from observed data. The ontology EABS does not perform reasoning on events since the event patterns are keeping out of the ontology instead of including them as ontology

rules.

Devaraju et al. developed a Sensing Geographic Occurrence Ontology (SEGO), which represents relations between geographic events and sensor observations and an application ontology for blizzard disaster (Devaraju et al., 2015). SEGO does not provide information about the environmental domain, but instead, it is developed as a starting point for the construction of application ontologies to infer geographic events from sensor observations. SEGO is based on DOLCE foundational ontology and is centered around the sensing domain. Therefore, Devaraju et al. distinguish stimulus from environmental processes to emphasize the process that actuates a sensor to produce observations.

In Table 2.1, we present a comparison of the different studied approaches using ontologies to solve semantic heterogeneity in the environmental monitoring domain. The comparison is based on the principles of ontology development's main criteria. We selected four comparison criteria based on a literature review: building purposes, domains, methods for knowledge acquisition, and links to other ontologies (Fernández-López et al., 2019).

Table 2.1: Comparison of existing ontologies related to EO.

Ontology	Knowledge acquisition method	Domain	Links with other ontologies	Building purpose	Literature
Environment Analytics domain ontology	Extract terms from real-world use case.	Environment Analytics domain	QUDT, SSN, GeoSPARQL	Help oil and gas companies to monitor and reduce the environmental impact of their operations.	(Bou-Ghannam, 2013)
EABS and flood monitoring ontology	Not mentioned	In situ sensor observations	SSN, DOLCE	Infer events from in situ observed data	(Llaves and Kuhn, 2014)
SEGO and blizzard ontology	Not mentioned	Geographic events	SWRL Temporal Ontology, DOLCE	Infer geographic events from sensor observations	(Devaraju et al., 2015)
ENVO	Extend ENVO original version with environmental materials, geographic features, etc.	Environment, biomedicine, ecology, food, habitats, and socioeconomic development domains	BFO	Delineate the environment domain	(Buttigieg et al., 2016)
GCI environmental ontology	Extract terms from the international standard ISO37120	Environmental indicators	GCI foundation ontology, SSN	Represent ISO37120 Environmental theme indicators	(Dahleh and Fox, 2016)
MDO	Analyzing the existing research work and disaster-causing mechanism	Meteorological disasters	No link to existing ontology	Describe relations between the components of meteorological disaster	(Zhang et al., 2016)
WDTF ontology	Extract terms from WDTF documents	Water observation domain	No link to existing ontology	WDTF data validation	(Shu et al., 2016)
Model ontology and data ontology	Building an abstraction flood workflow	Flood disaster management	No link to existing ontology	link environmental models with disaster-related data to support flood disaster management	(Qiu et al., 2017)
WaWo+	Domain experts expand the original WaWO ontology with water quality indicators, urban water terms, and actors.	urban water resources	domain WaWO ontology	Data and knowledge interoperability and data integration to facilitate urban water resources management within a river basin.	(Oliva-Felipe et al., 2017)
SemsorGrid4Env ontologies	Not mentioned	fire risk monitoring, coastal and estuarine flood	SSN, SWEET, the Ordnance Survey ontologies	Sensor data integration.	(Gray et al., 2009)
Monitor Ontology	The collection and efinition of a common base vocabulary in the thematic fields of MONITOR	Risk domain	DOLCE	monitoring methods and environmental risk management.	(Kollarits et al., 2009)

2.3.4 Synthesis

During our literature review, we studied several ontologies developed in the field of earth observation and environmental monitoring. Despite these numerous works, several limitations can be noted. First, many studies focused only on the knowledge expression of a specific type of environmental observation (such as in-situ) or on specific requirements for a particular organization. Second, some existing ontologies are dealing with specific aspects of the environment. The environment is lying on multidisciplinary fields, such as meteorology, hydrology, geology, geography, and so on. Existing ontologies do not cover all environmental fields such as the ontology MDO neither the observation conditions/contexts such as the ontology ENVO, which does not include the sensing nor the spatiotemporal contexts. Accordingly, the coverage to annotate or link observed data will not be possible with these limited knowledge ontologies. Third, little attention was paid to the semantic relationships between different environmental components such as disasters, observations' contexts in time and space, and other correlated environmental conditions that can provide a comprehensive view to analyze environmental factors. To address this issue, some ontologies were built from specific case studies such as the Environment Analytics domain ontology and GCI environmental ontology. However, this kind of ontology cannot model environmental data sources whose information is not present in the specific case study. Finally, existing monitoring systems generally use a specific reasoning method on specific data sources and formats to generate inferences, which makes it difficult or even impossible to obtain inferences across heterogeneous data sets.

Our objective is to address the listed gaps by building a domain ontology for environmental monitoring in order to link and integrate observed data from various sources. The proposed ontology will aim at (1) ensuring semantic interoperability between heterogeneous data sources, (2) supporting knowledge discovery and generation, (3) integrating and linking data across various disciplines, and (4) providing a global information view.

2.4 State of the art of semantic data integration approaches

In the literature, various semantic data integration approaches have been proposed. These latter generally fall into two primary categories, depending on the type of data access: materialized and virtual data integration approaches. In what follows, we scrutinize related works and systems for each category.

2.4.1 Materialized approaches

The materialized data integration approaches consist of the static transformation of distributed data into a data warehouse or a single RDF store. In (Patroumpas et al., 2014), Patroumpas et al. proposed an ETL Tool for transforming geospatial data into RDF Triples. The so-called TripleGeo can extract and transform geospatial features from many input formats. It is based on Geometry2RDF and uses the WGS84 vocabulary and several geometric

types of GeoSPARQL. However, this tool requires a centralized data repository and significant manual effort to define the ETL rules to set up this repository.

Kyzirakos et al. proposed the open-source tool GeoTriples for transforming geospatial data from their original formats into RDF (Kyzirakos et al., 2018). This semi-automatic system enables the transformation of geospatial data stored in raw files (shapefiles, CSV, XML, etc.) and spatially-enabled RDBMS (PostGIS and MonetDB) into RDF graphs. This tool is implemented as an extension to the D2RQ platform and uses well-known geospatial vocabularies, e.g., the vocabulary of the OGC GeoSPARQL standard. It generates and processes R2RML and RML mappings that transform geospatial data from many input formats into RDF. The generated mappings may be revised, e.g., to utilize a different vocabulary. However, this step can only be done manually.

The main limitation of these two tools is that they deal only with geospatial vocabularies, including the OGC standard GeoSPARQL and does not allow users to select other semantic vocabularies or ontologies.

Abbes et al. proposed an approach based on MongoDB, a NoSQL database, and modular ontologies for ontology-based Big Data integration (Abbes and Gargouri, 2018). This approach follows three steps: 1) wrapping data sources to MongoDB databases, 2) generating local ontologies by learning ontology from MongoDB, and 3) merging the local ontologies to get a global one. To this end, they choose MongoDB, a document-oriented database ontology, to address the variety dimension of Big Data. However, the step of wrapping each data source in the MongoDB database is expensive in terms of time and storage. Moreover, this approach requires several efforts to include an additional data source. Specifically, it is necessary to define a local ontology for the new data source.

Bansal et al. introduced a semantic ETL framework that uses semantic technologies to integrate and publish data from multiple sources (Bansal and Kagemann, 2015). The proposed framework generates a semantic model to integrate heterogeneous datasets. Then it generates semantic linked data in compliance with the data model. This generated semantic data is made available on the web as linked data (RDF triples) available for querying and used in analytics and data-driven innovative apps. The use of semantic technologies is introduced in the transform phase of the ETL process to generate RDF triples, which will be stored in a data mart or warehouse. However, this framework involves a manual process of analyzing the data sets and their schema.

Aarnio et al. proposed an ELT system to support condition-based monitoring in automation systems (Aarnio et al., 2014). The approach consists of four-steps of transformation process from local data to RDF. Firstly, an automatic transformation of source data from local source formats to temporary RDF data is done. Then, the temporary RDF data is transformed into instances of local ontologies, where the local ontologies should conform to shared vocabularies. The third step links between data from local ontologies. Finally, rule sets are developed and executed on top of local ontologies to infer new information. A case knowledge base is then generated with an inference engine to support data access using SPARQL queries. However, the initial development of this system involves considerable effort. Experts need to

develop local ontologies for each data source based on a defined (or already available) shared vocabulary. Then, it also needs efforts to generate a lot of mappings (alignments) between local and global ontologies using semantic relations.

Knoblock et al. proposed the Karma Web system, a semi-automatic data modeling and integration framework that can integrate data from a variety of sources, including relational databases, spreadsheets, XML, JSON, and RDF files (Knoblock et al., 2012). Karma consists of four steps: 1) Importing data from sources, 2) Cleaning and normalizing data, 3) Building a model, which can be an ontology, for each source. Finally, 4) Integrating data using this model and loaded it into an RDF store. The tool learns the model that has been used to publish RDF data so that the semantic type need not be added every time. The model is set up by the user by importing an ontology into the system. Rather than Karma quickly integrates data by allowing automatic modeling by learning, the entire integration process needs a lot of human intervention in cleaning and modeling while choosing classes and relations from ontologies.

Rani et al., have proposed MOUNT, a multi-level annotation and integration framework (Rani et al., 2019). This approach incorporates three phases, such as a coarse-grained annotation, a fine-grained annotation, and query processing. In the coarse-grained, Yago ontology and SESE, a keyword-based ontology search engine, are used for categorizing the domain information of the heterogeneous data sources. The fine-grained annotation handles the unified data integration of structured and unstructured data. With the help of multi-level semantic annotation, the MOUNT approach creates the global RDF ontology to ease the query process. Despite the good experimental results, the loss of information related to the use of this approach can have negative consequences for the decisions made. Also, this framework needs to be improved in terms of response accuracy.

2.4.2 Virtual approaches

As described in Section 1.3, virtual data integration approaches do not copy the original data. Instead, they virtually integrate the data in a way that preserves the sources' autonomy, and at the same time, allow for a distributed processing of queries. In the following, we analyzed several systems based on virtual data integration approaches, while distinguishing between mediator-wrapper-based systems and federated systems.

2.4.2.1 Mediator-wrapper-based systems

In the context of semantic virtual data integration, a mediator-wrapper-based system mainly corresponds to the OBDA approach. The integrated schema is represented as an ontology, and the data sources generally correspond to RDBMSs, but some research has also been conducted to tackle XML, CSV, and NoSQL stores by choosing the appropriate wrappers. Indeed, wrappers act as a bridge between client applications and heterogeneous data sources. They receive user requests in the form of SPARQL queries, translate them into the query

languages of the respective data sources, and return the results after query execution (Curé and Blin, 2014).

Ontop is an open-source OBDA system that allows for querying relational data sources (Calvanese et al., 2017). This system exposes relational databases as virtual RDF graphs by linking the classes and the relationships in the ontology to the data sources through R2RML or OBDA mappings. The virtual RDF graphs can then be queried using SPARQL. The core of Ontop is the SPARQL engine Quest, which is in charge of rewriting SPARQL queries over the virtual RDF graph and ontology into SQL queries. A drawback of the Ontop framework is its limitation of accessing only a single relational database and does not support other data formats. Additionally, this database has to be SQL compliant (Steindl and Kastner, 2019).

In (Bereta et al., 2018), the authors proposed an OBDA technique to deal with geospatial data coming from different sources. It is implemented in the system Ontop-spatial, which enables the integration of vector or raster data stored in geospatial relational databases using ontologies and mappings. Nevertheless, this system only deals with geospatial vocabularies and does not support data formats other than geospatial data stored in relational databases.

In (Kharlamov et al., 2017), the authors presented the Optique platform, an Ontology-Based Stream-Static Data Integration (OBSSDI), a paradigm that extends classical OBDA for direct end-user access to streaming, historical, and static data. The platform allows integrating relational databases and data streams based on mappings created through imported ontologies. It also supports the formulation of queries using STARQL, a specific query language that natively supports OBSSDI hybrid queries. The queries produced by the STARQL translator are processed and answered by Optique's dedicated data stream management system, ExaStream. ExaStream has been designed for efficiently processing on both static and streaming information and the corresponding queries produced by the STARQL engine. However, this approach is developed on the basis of Siemens requirements to enable direct access to data generating from service centers for power plants. Besides, data formats supported by Optique are relational databases, streaming, and sensor data.

For the purpose of integrating heterogeneous data formats, other mediator-wrapper-based systems designed several wrappers, where each one is used to access data of a specific format. In (Adeyelu and Anyebe, 2018), the authors developed a mediator-wrapper approach to semantically integrate heterogeneous databases based on the LAV paradigm of data integration. The mediator provides a virtual database, i.e., a global schema, in the form of classes to the applications and the wrappers. When an input query is posted to this global schema, the mediator uses a global ontology to know which wrapper to call. The wrapper converts the input query into a local query that its data source can understand. Afterward, it executes the query and gets the result. The result that it receives is converted to a format understandable by the mediator. To do so, each wrapper uses a local ontology. Admittedly, the system facilitated easy access to information from different databases of diverse platforms. However, it is necessary to define a local ontology for each wrapper, in a sense, for each new data source. Moreover, data sources that could be integrated into this system are only relational databases. Furthermore, since this system adopts a LAV-based semantic mapping approach, each modification in the global schema needs to change all of the local ontologies.

The Semantic Web Integrator and Query Engine (SemWIQ) also uses an architecture based on the mediator-wrapper approach (Langegger et al., 2008). It provides access to distributed data sources using SW technology. The proposed system allows retrieving data using SPARQL queries. It consists of a mediator which is accepting queries from clients and then collects data translated by a number of wrappers attached to local data sources. The mediator analyzes an input query and scans the catalog to identify relevant data sources. SemWIQ uses VoID (Vocabulary of Interlinked Data sets) as a data catalog [2]. The resulting plan is executed by sending subqueries to the sources. Later on, the wrappers use mappings to translate data from the underlying information systems into a common schema that can be processed by the mediator. However, in such kind of systems, any data source must use a SPARQL-capable wrapper to provide local data.

The authors in (Regueiro et al., 2017) proposed a framework for the semantic mediation between environmental observation datasets through OGC SOS interfaces. The design of the framework is based on mediator-wrapper architecture. Each wrapper is designed explicitly for the characteristics of a data source, and it adapts its specific data model and data access interface to O&M and SOS. As a basis for the specification of data integration knowledge, the system uses SSN and SWEET ontologies. The Apache Jena SPARQL engine ARQ was used to query the mediator. Unfortunately, this framework does not specify the query processing either optimization, and its main drawback is the development of new wrappers for new application domains with different data sources.

2.4.2.2 Federated systems

The second variant of virtual data integration approaches is federated systems. As described in Section 1.3 and based on survey results (Oguz et al., 2015) (Saleem et al., 2016), we focus on the most popular federated systems where the data is accessed through SPARQL endpoints—this is the case for the FedX, Splendid, and DARQ systems. These systems differ in the mechanisms used in the different steps of the distributed query processing (source selection, query rewriting, query planning, and query evaluation). One of the most important optimization steps in federated SPARQL query processing is the efficient selection of relevant sources for a query. In the following, we give an overview of existing SPARQL query federation systems with respect to source selection type.

FedX is a framework developed by FluidOps for transparent access to distributed sources through a query federation and is available as open-source software (Schwarte et al., 2011). FedX performs source selection without prior knowledge of sources. Only SPARQL ASK queries are sent for each triple pattern to identify the relevant sources.

DARQ is another federated system that relies on service descriptions to identify relevant sources (Quilitz and Leser, 2008). The service descriptions consist of information on predicate capabilities and statistics on triples hosted in each source. They have to be declared in advance before query processing to decide where a sub-query should go.

[2]VoID Vocabulary: https://www.w3.org/TR/void/

Splendid is a SPARQL endpoint federation system that makes use of VoID descriptions as a data catalog along with SPARQL ASK queries to perform the source selection step (Görlitz and Staab, 2011). VoID takes the form of an RDF schema that supports the description of metadata about RDF data sets.

The Hibiscus system showed better results in terms of source selection and runtimes than the aforementioned systems (Saleem and Ngomo, 2014). Its approach is based on modeling SPARQL queries as directed-labeled hypergraphs. Based on this representation, algorithms have been designed to discard sources that are not pertinent to the computation of the final result based on the types of joins present in the query.

MULDER is another SPARQL query engine for federated access to SPARQL endpoints (Endris et al., 2017). It describes data sources in terms of RDF molecule templates, i.e., abstract descriptions of entities belonging to the same RDF class. It mainly exploits those descriptions to select the datasets that can increase the completeness of the answer.

2.4.3 Synthesis

Table 2.2 summarizes the comparison of the different semantic data integration approaches, mentioned above in terms of seven criteria:

- **Data acquisition:** We identified five mainstream approaches for acquiring data from local sources, namely ETL, ELT, OBDA, mediator, and federation.
- **Data access:** data integration approaches provide two ways to access data; direct access to the centralized repository (generally in materialized approaches) and access to the mediator/federator (or the shared vocabulary in the virtual approaches).
- **Semantic vocabulary:** to identify the ontology or the semantic model used to integrate data.
- **Data types:** The primary focus of a data integration approach is on the data types, including relational databases and geospatial.
- **Mapping complexity:** reflects the complexity of relations between data sources and the semantic vocabulary (the ontology). This characteristic depends on the differences in data integration approaches capabilities to represent mappings.
- **Query processing type:** The focus of most data integration systems is on querying disparate data sources. This field identifies the type of the query processing mechanism (centralized / distributed) if the data integration system includes one.
- **Source selection:** The among primary focuses of a query processing approach is the identification of relevant data sources that are likely to contribute to the final result. There are mainly two methods: metadata catalogs source selection and sampling query-based source selection (SPARQL ASK queries).

Table 2.2: Comparison of existing semantic data integration approaches.

Approach	**Data acquisition**	**Data access**	**Semantic vocabulary**	**Data types**	**Mapping complexity**	**Query processing type**	**Source selection technique**
MOUNT (Rani et al., 2019)	ETL	Query the generated RDF ontology	Create a domain ontology using Yago Ontology and SESE	Structured and unstructured data.	Create mappings between all the source schemas and define rules to convert metadata into RDF graph.	Centralized	Not included
Geotriples (Kyzirakos et al., 2018)	ETL and OBDA	Access to the generated RDF data stores, or querying over virtual RDF graphs defined through R2RML mappings.	OGC GeoSPARQL standard	Geospatial data (shapefiles, CSV, XML, and spatially-enabled RDBMS).	Semi-automatic mappings generation. To use a different vocabulary, generated mappings may be manually revised.	Centralized and distributed	Not included
(Abbes and Gargouri, 2018)	ETL	Query the global ontology	Create a local ontology for each data source.	Data formats are able to be transformed into a NoSQL database.	- Define rules to generate the corresponding MongoDB database to a data source and mapping rules to extract local ontology's concepts and relations from MongoDB constructs. - Syntactic and semantic similarity measures to integrate local ontologies in a global one.	Centralized	Not included
Ontop-spatial (Bereta et al., 2018)	OBDA	Access virtual RDF graphs defined through R2RML mappings.	OGC standard GeoSPARQL	geospatial databases	Ontop-spatial supports two mapping languages: R2RML and the native Ontop mapping language	distributed	Data (tables) catalog
(Adeyelu and Anyebe, 2018)	mediator	wrappers	A domain ontology	Relational databases	LAV approach	distributed	Not included
MULDER (Endris et al., 2017)	federated	SPARQL endpoints	Any linked open vocabulary	RDF stores	Not mentioned	distributed	describing data sources in terms of RDF molecule templates

Table 2.2: Comparison of existing semantic data integration approaches.

Approach	Data acquisition	Data access	Semantic vocabulary	Data types	Mapping complexity	Query processing type	Source selection technique
Optique (Kharlamov et al., 2017)	OBDA	Access virtual RDF graphs	Siemens ontology	Relational databases, streaming and sensor data	semi-automatically bootstrap an initial ontology and mappings	distributed	Not included
Ontop (Calvanese et al., 2017)	OBDA	Access virtual RDF graphs defined through R2RML mappings.	OGC standard GeoSPARQL	Relational databases	Supporting two mapping languages: R2RML and the native Ontop mapping language	distributed	Data (tables) catalog
SOS semantic mediation framework (Regueiro et al., 2017)	mediator	wrappers	SSN and SWEET	Environmental observation datasets through OGC SOS interfaces	LAV approach Adapting the data model and data access interface of each source to O&M and SOS	distributed	Not included
Semantic ETL framework (Bansal and Kagemann, 2015)	ETL	Access to the generated semantic linked data stored in a data mart or warehouse	Create a semantic data model.	Linked data available on the web in flat file formats such as csv, xls, and txt or through a RESTful client.	If the data sources belong to disparate domains, multiple ontologies are required and alignment rules must be specified.	centralized	Not included
HIBISCUS (Saleem and Ngomo, 2014)	federated	SPARQL endpoints	Any linked open vocabulary	RDF stores	Not mentioned	distributed	Modeling SPARQL queries as directed-labeled hypergraphs
TripleGeo (Patroumpas et al., 2014)	ETL	Access on the centralized RDF data repository	OGC GeoSPARQL standard	Geospatial data (shapefiles and DBMS)	Manual effort to define the ETL rules.	Centralized	Not included
(Aarnio et al., 2014)	ELT	Direct access to local ontologies, and access to the shared vocabulary.	CBROnto ontology	Relational databases, RDF, Spreadsheets.	Automatic generation of the mappings between local and global ontologies using semantic relations.	Centralized	Not included
Karma (Knoblock et al., 2012)	ETL	Access to the generated RDF data store or through mappings.	An ontology as input	CSV, JSON, XML, RDF, relational databases	A lot of human intervention in modeling.	centralized	Not included

Table 2.2: Comparison of existing semantic data integration approaches.

Approach	Data acquisition	Data access	Semantic vocabulary	Data types	Mapping complexity	Query processing type	Source selection technique
Splendid (Görlitz and Staab, 2011)	federated	SPARQL endpoints	Any linked open vocabulary	RDF stores	Not mentioned	distributed	VoID and SPARQL ASK queries
FedX (Schwarte et al., 2011)	federated	SPARQL endpoints	Any linked open vocabulary	RDF stores	Not mentioned	distributed	SPARQL ASK queries
SemWIQ (Langegger et al., 2008)	mediator	SPARQL-capable Wrappers	Any linked open vocabulary	Various types (RDF, relational databases, etc.)	Not mentioned	distributed	Data catalog
DARQ (Quilitz and Leser, 2008)	federated	SPARQL endpoints	Any linked open vocabulary	RDF stores	Not mentioned	distributed	service descriptions

As described above, studied research works fall into two primary classes: virtual and materialized data integration approaches. The materialized approach is the static transformation of distributed data into a data warehouse or a single RDF store. That's why it is also called the Extract-Transform-Load (ETL) approach. The main advantage of this approach is that query processing is centralized and fast; the integrated data is gathered into a single source; thus, a rewriting phase of queries is no necessary.

However, these materialized approaches have major disadvantages, including the need of extra storage since data is replicated (Benhlima and Chiadmi, 2006), this implies an extra cost of storage, as well as the cost of maintenance (Chen et al., 2004). Furthermore, the materialized data may rapidly be outdated if the data source is frequently updated. A workaround is to run the data extraction and transformation processes periodically. But, in the context of huge datasets, a compromise must be found between the cost (in time, memory, and CPU) of materializing and the freshness of the large scale of available data. Alternatively, the virtual data integration approach was proposed to keep data in their original sources and access them on-the-fly using a query language. Such approaches solve the problem of the real-time integration of heterogeneous data and have the advantage of avoiding the cost of materialization. Virtual data integration approaches mainly focus on the techniques of queries processing; thus, they are also known as Distributed Query Processing (DQP) approaches.

Despite the numerous reviewed virtual data integration works, several limitations can be noted. Actually, many mentioned approaches lack automation. They require a lot of human intervention, particularly in data mapping and modeling, and the semantic annotation process is mostly done manually by human annotators. Moreover, most of the approaches are dealing with particular data formats (relational databases for Ontop, for example). Although recently proposed approaches aim to integrate heterogeneous data formats, there is still a lack of schematic and semantic interoperability among the different data sources.

As outlined before, virtual data integration systems can be divided into mediator-wrapper-based systems and federated systems. In the former, the use of wrappers facilitates the integration of distributed data sources providing heterogeneous formats (databases, CSV, XML, etc.). However, several wrappers need to be written to convert the query into each data source's format and then convert the returned result into the common format of the mediator. Furthermore, wrappers are specific language-capable. For instance, in case the global schema is defined as an ontology, the data integration system should implement SPARQL-capable wrappers. Accordingly, if the language of the global schema changes, all wrappers should be updated.

In the case of federated systems, the implementation of wrappers is not required since the data sources involved in the federation support the global schema and the query language that the federation agreed upon. Nevertheless, there is a need to specify target endpoints for each data source.

Regardless of the architecture, the source selection step was variously performed in virtual data integration systems. We classified the data source selection methods according to which need metadata catalogs and those which do not. These latter can be summed up in

the SPARQL ASK queries method. However, query processing systems show that using a metadata catalog is a need, since omitting it makes source selection quite difficult and time-consuming due to the need for preprocessing of each query. This time-consuming task is usually done by SPARQL ASK queries (Oguz et al., 2015). The reviewed metadata catalogs can be classified into three types: (i) service descriptions, (ii) VoID (Vocabulary of Interlinked Data sets) descriptions, and (iii) list of predicates. The service descriptions provide metadata information as well as some statistics about the data (such as the number of triples containing a predicate and total triples). In addition to the information about datasets, VoID allows the description of RDF links between them, called linkset. However, this linkset can only be defined because datasets are RDF stores. Consequently, describing an RDF link between datasets consists of creating RDF triple whose subject and object are expressed in different datasets. The last metadata catalog only provides a list of predicates for each source since the number of predicates is less than the number of subjects and objects. The predicate list can be used to decide the destination source for each sub-query pattern. Nevertheless, existing source selection techniques do not identify the hidden relationships in the data as long as they do not explore the high semantic expressiveness of domain ontologies.

2.5 Discussion and orientations

To overcome the aforementioned limitations of the reviewed data integration approaches, we will adopt a hybrid architecture, benefiting from both mediator-wrapper-based and federated approaches. Our system architecture should rely on mediator-wrapper based architecture to accommodate different kinds of data sources, but rather than implementing wrappers, a set of generated RDF stores will be queried like federated systems. Furthermore, to enhance the source selection, the query processing needs to move beyond the discovery of simple one-to-one equivalence matches to the identification of more complex relationships across datasets and extracting the sources that could contribute to the response of an input query. For that purpose, data should be organized in a manner so that experts can easily understand it, extract information from, and mainly infer implicit knowledge that improves the understanding of the data. The purpose is to provide a common information space, where different EO systems and programs will have unhampered and uniform access to the available environmental data that will be linked and integrated into a single knowledge graph. This global information view will allow the data sources to speak the same language and to share information so that domain experts and software agents could transform them into actionable knowledge. We refer here to a knowledge graph (Ehrlinger and Wöß, 2016). Accordingly, instead of structuring the metadata catalog as a collection of relational tables or descriptors, the Virtual Knowledge Graph (VKG) was proposed to replace the rigid structure of tables with the flexibility of graphs that are kept virtual and embed business knowledge (Xiao et al., 2019). A VKG is a new paradigm for data integration and access that inherently exploits data virtualization and that in addition, overcomes the difficulties of traditional approaches based on the relational model.

Using an ontology makes it simpler for users to formulate their information needs, which

are then automatically translated into a query over the data sources. However, it seems to be beneficiary and feasible, elaborating a method that focuses on data and their relationships and places them into different contexts, especially given that many relationships in the EO domain are more complicated than what can be represented in graphs. That's why we need an extra layer to further link the data in order to extract new knowledge without accessing materialized data. As outlined earlier, we believe that the adequate mathematical formalism that is capable of representing complex relationships is hypergraphs since they generalize graphs by allowing edges to connect more than two nodes, which may facilitate a more precise representation of environmental knowledge. They have the advantage of reducing the degree of complexity and variety of relationships between models and can be applied to the information and data model. Accordingly, relying on the virtual data integration framework, we propose to add a Virtual Knowledge HyperGraph (VKHG) layer above the mappings layer in order to semantically link the mappings using the domain ontology. Figure 2.3 shows the difference between a traditional virtual data integration framework and our orientation to propose a VKHG-based data integration framework.

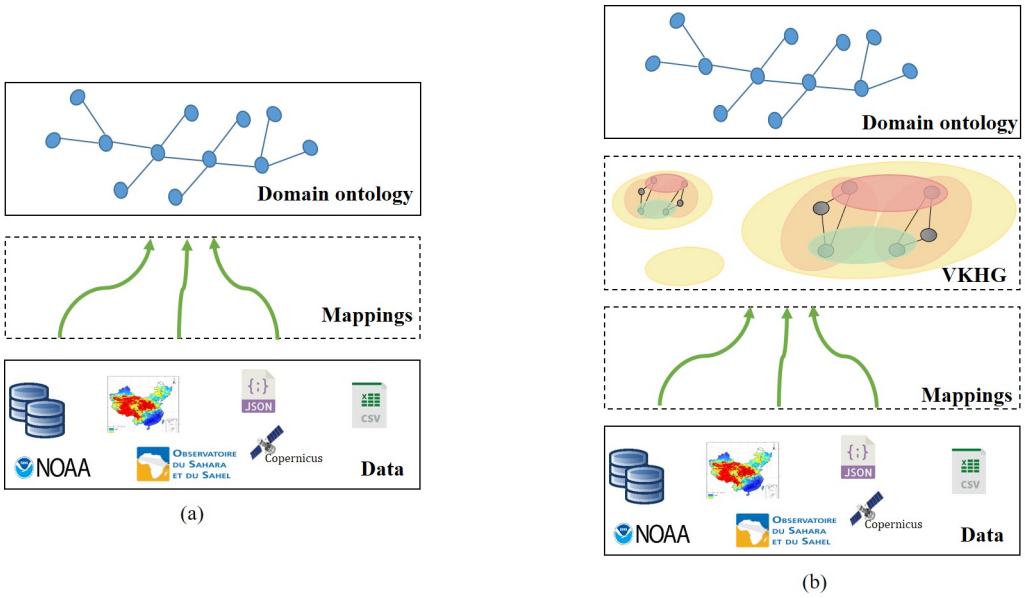


Figure 2.3: (a) Traditional virtual data integration framework (Zidane et al., 2018). b) Our orientation to VKHG -based data integration framework.

Accordingly, the query processing approach will mainly address source selection by taking advantage of hypergraphs. Our innovation consists of using hypergraphs to improve query engine performance in terms of result completeness. The query processing will aim at adapting the returned results and makes them more precise, more relevant, and richer in terms of semantic relationships.

To summarize, our orientation in this research is to propose a semantic data integration approach;

- that is virtual,
- that adapts a hybrid architecture between mediator-wrapper and federation (see Figure 2.4),
- overlapping a VKHG layer to the virtual data integration framework,
- and aiming at improving source selection to enhance query processing in terms of completeness and response relationship richness.

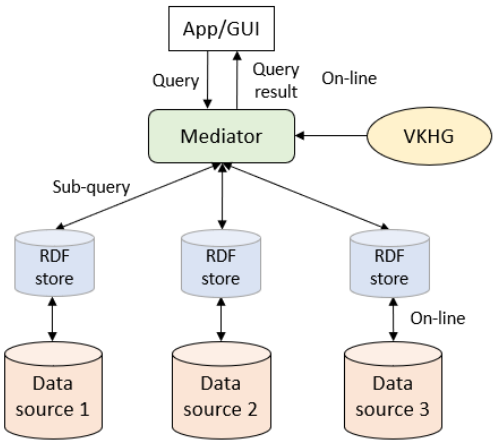


Figure 2.4: Proposed hybrid query processing architecture.

2.6 Conclusion

In this chapter, we discussed the related works concerning ontologies related to EO domain, semantic data integration, and query processing. Then, we presented the main orientations of this thesis. Specifically, we first presented a review of existing ontologies related to the EO domain, including sensor ontologies, observation & measurement ontologies, and environmental monitoring ontologies. Next, we reviewed the existing approaches for data integration under two main categories: materialized and virtual. We also discussed source selection techniques in query processing, which is a fundamental task when developing a virtual data integration system. The outcomes of the state-of-the-art acts as a stepping stone towards the proposal of a new semantic data integration approach. The main contributions of this thesis will be presented in the following chapters.

Chapter 3

MEMOn: Modular Environmental Monitoring Ontology

Sommaire

3.1 Introduction 59
3.2 Ontology development methodology 59
3.3 MEMOn development process 60
3.3.1 Exploration phase 61
3.3.2 Planning phase 62
3.3.3 Module development phase 65
3.3.4 Release phase 86
3.4 Conclusion 95

3.1 Introduction

With the wide variety of environmental data, it is becoming increasingly difficult for domain experts to understand natural phenomena and reduce the adverse effects of climate change. The exploitation of EO data is still limited due to the silos between data sources, systems, and programs. Each source offers a different data schema or model to represent domain knowledge that resides in the experts' minds. Consequently, data analysts need to establish contact with original data sources and model producers to understand and use them properly. To deal with these issues, in this work, we aim to conceptualize and formalize a modular ontology for environmental monitoring, called MEMOn, in order to provide a common, shared vocabulary of the environmental domain that will be exploited later to semantically link and integrate EO data from different sources through a knowledge graph. In this chapter, the different steps of the development process of MEMOn are presented.

3.2 Ontology development methodology

Building an ontology is not a straightforward task, especially when ontologies become more significant and more complex (Corcho et al., 2004). Therefore, a methodology that guides

and manages the ontology development is necessary. But, one of the thorniest problems is how to choose the appropriate methodology taking into account the complexity of the domain to be modeled and the furthest evolution and reuse. To support ontologies building, several methodologies, such as METHONTOLOGY, NeON, and OntoClean, have been discussed in (Karray et al., 2012). These methodologies can be classified according to two criteria: agility and modularity. Agile methodologies enable the incremental and iterative development of an ontology. Otherwise, modular-oriented methodologies consider the modularization aspect in the ontology development process. In this work, we adopt the AOM (Agile methodology for developing Ontology Modules) methodology (Gobin, 2013). This choice is argued by the fact that AOM is an agile modular-oriented methodology; it will enable the incremental and iterative development of a modular ontology's modules. Whereas, other agile methodologies are suited to develop ontologies in one go.

AOM stipulates the sequence of four phases: exploration, planning, module development, and finally, release phases. We adjusted these phases according to our needs and contexts, as shown in Figure 3.1. More specifically, we added a "Global CQs (Competency Questions)" step to facilitate the extraction of stories (scenarios) related to the environmental domain.

Accordingly, the development life-cycle starts with the exploration phase, in which we identified the ontology requirements and objectives, with the help of domain experts. Then, we defined a set of generic CQs according to the studied domain and experts' needs. The competency questions are defined as a set of questions stated in natural language so that the ontology must be able to answer them correctly (Grüninger and Fox, 1995). Given this set of generic CQs, we extract stories to represent the different contexts that limit the target domain (in the planning phase). Indeed, stories are scenarios related to specific contexts from which facts used to build the ontology can be obtained. These stories are written based on the information gathered during interview sessions with domain experts. Then, the ontological modules are defined based on these stories. In the module development phase, a set of specific CQs related to one module is defined then used to design a semi-formal module. Then, the modules are formalized, implemented, and evaluated to be merged in the release phase. The next sections detail these four phases, according to the adapted AOM Methodology.

3.3 MEMOn development process

The so-called MEMOn (Modular Environmental Monitoring Ontology) aims to semantically capture the knowledge of the environmental monitoring domain. In this section, we present in detail the different steps of the development process (exploration, planning, development, evaluation) of MEMOn.

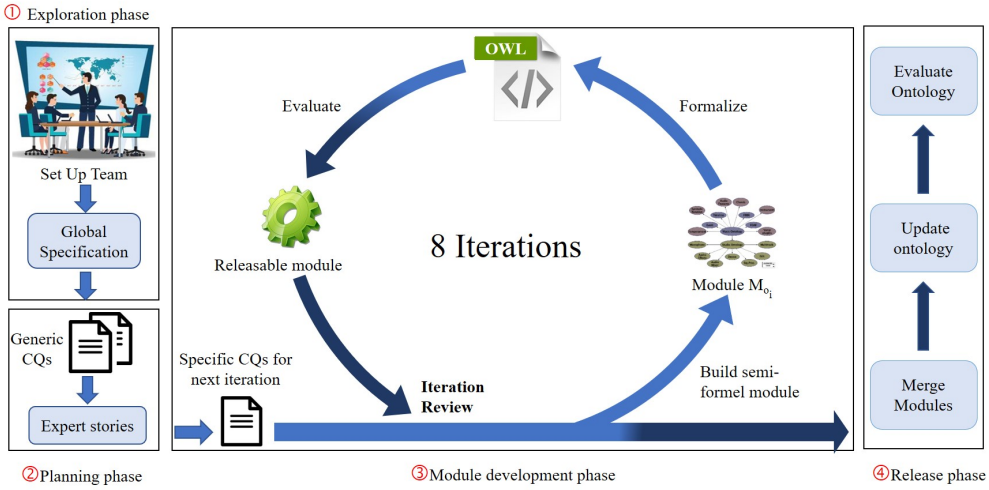


Figure 3.1: The adapted AOM development life-cycle.

3.3.1 Exploration phase

The ontology development process starts with the exploration phase, whereby the domain expert is part of the development team, and with the knowledge engineer, both embark on the development process. To have a better overview and understanding of the domain of environmental monitoring, we refer to OSS (OSS, 2010), our project partner, which has significant expertise in environmental monitoring, natural resource management, and climate change. During this phase, the ontology engineers and the OSS experts sat together and discussed the domain of interest, the objectives and the requirements of the ontology.

3.3.1.1 MEMOn objectives

The objective of MEMOn ontology is to provide a formal representation of the domain of environmental monitoring and encapsulate the knowledge contained in multiple environmental data sources. MEMOn will support:

- semantic interoperability,
- knowledge discovery and generation,
- data integration and linking,
- and a global information view to better understanding environmental phenomena.

3.3.1.2 MEMOn requirements

Today, so many ontologies and knowledge repositories have been developed. However, many problems are encountered when knowledge engineers and users want to understand and reuse the ontologies into their own applications. Ontology modularization can, to a certain extent, help to solve this problem, since it contributes to maintaining the clarity and the coherence of the ontology by presenting ontology modules with the needed knowledge. There are two approaches to ontology modularization (Abbes et al., 2012). The first approach concerns the extraction of modules from a large ontology, whereas the second approach is the integration of several modules that have been developed separately to form a new large ontology with respect to a specific domain. In this work, we focus on the second approach by developing separate ontological modules to build a global ontology that captures the knowledge related to the environmental monitoring domain.

However, ontologies developed independently, remain in almost every case unmapped. Moreover, even where mappings are created, they are rarely able to be updated in the needed ways, because the ontologies on both sides of the mapping relation are themselves being changed independently and on asynchronous cycles. Accordingly, the use of upper-level ontologies facilitates the alignment between several ontological modules. In this work, the use of an upper-level ontology is fundamental to ensure interoperable reuse and integration of the proposed ontological modules. Furthermore, reusing classes from mid-level and domain ontologies is also a possible solution to facilitate ontology construction. In fact, ontology reuse has several advantages. First, it is no longer necessary to build the ontology from scratch. Second, it increases the quality of new ontologies because the reused components have already been tested. Finally, ontology reuse improves the efficiency of ontology maintenance (Ding et al., 2007).

To summarize, the key requirements of MEMOn are listed as follows:

- The ontology captures the notions related to the domain of environmental monitoring.
- The ontology applies the principles of modularization.
- The ontology is aligned with an upper-level ontology.
- The ontology reuses classes from mid-level ontologies that specialize classes in the upper-level ontology and other domain ontologies.

3.3.2 Planning phase

During the planning phase, generic CQs are defined to cover all needed information mentioned in the domain knowledge. To do so, we start by exploring the knowledge about the environmental monitoring domain, discovered and extracted in coordination with the domain expert (OSS). In addition, we elicit additional knowledge about the domain of interest by exploring multiple data sources, including OSS, Copernicus, and EMDAT (EM-DAT, 2012).

Each source has its own set of relevant data in specific disciplines. For example, OSS generated climatological data (data about temperature, precipitation, etc.), whereas, EMDAT database provides data about natural disasters.

Based on MEMOn domain knowledge, we define a set of generic CQs, which the ontology should be able to answer. In the following, some examples of the defined generic CQs are presented in Table 3.1.

Given this set of generic CQs, stories are extracted to define the perimeter of the domain of study. So, we identified four main stories which contribute to environmental monitoring that MEMOn will consider (see Figure 4.2):

1. **Environmental conditions story** which includes geological structures (such as rock and soil types), geographical features (e.g., forest, land), hydrological structures (e.g., water, oceans, rivers, lakes), different kinds of natural disasters, and the corresponding environmental processes (such as ground shaking).
2. **Urbanization story** which includes different kinds of infrastructures,
3. Observing conditions story which includes sensing systems (sensors, satellites and so on) and observation conditions (properties and measurement conditions),
4. **Spatiotemporal conditions story**; Observations, disasters, and environmental processes occur in specific spatiotemporal regions. Hence, MEMOn represents the times and locations of environmental phenomena or observations.

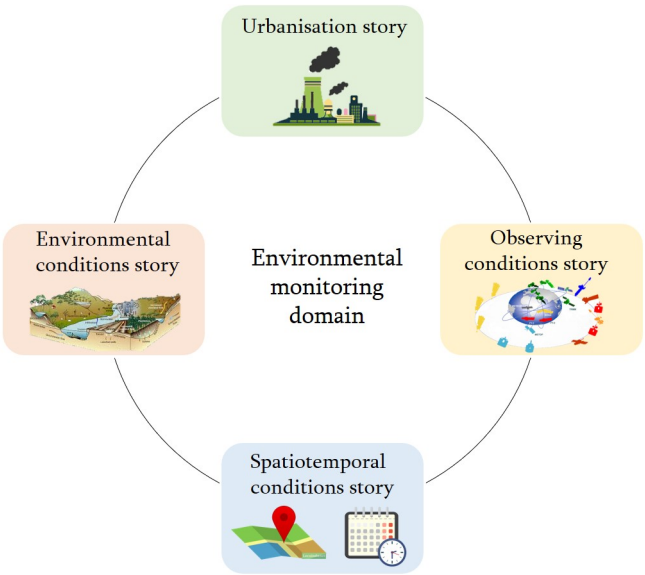


Figure 3.2: Environmental monitoring domain' expert stories.

After reviewing the different standards and ontologies proposed in the domain of environmental monitoring, as presented in chapter 2, we propose eight ontological modules from the

Table 3.1: Competency questions examples.

CQi	Competency question
1	What is the value of a given observation in a region [X] in date [Y]?
2	What are the measurements for an observed property [X] for natural disaster [Y]?
3	What is the average measurement for climate parameter [X] during natural disaster [Y]?
4	What is the measurement unit used for a specific property [X]?
5	What are the observations associated with a specific environmental process [X]?
6	What is the type of sensor used for a given observation [X]?
7	Which natural disasters may lead to natural disaster [X]?
8	What is the disaster resulting from an environmental process [X]?
9	What are the environmental processes that caused the natural disaster [X]?
10	Where did the environmental process [X], which is associated with natural disaster [Y], take place?
11	What is the environmental process [X] that is preceded by the environmental process [Y] during natural disaster [Z]?
12	What are the environmental materials involved in the environmental process [X]?
13	What environmental material is the output of the environmental process [X]?
14	Where did the natural disaster [X] take place?
15	What are the spatial zones where a specific disaster activity is the highest?
16	Is the infrastructure [X] implicated in the occurrence of the natural disaster [Y]?
17	When did the natural disaster [X] take place?
18	What is the environmental process that precedes the environmental process [X]?
19	Is the environmental process [X] has end location [Y]?
20	What is the environmental process that co-occurs with the environmental process [X]?

identified stories. The first story deals with *environmental materials*, *environmental processes*, and *natural disasters*, regardless of the discipline (hydrology, geology, meteorology, etc.). So, we identify them as three modules. The second story results in the *infrastructure module*. From the third story which handles the observations, their measurements, and systems that produce them, we identify two modules: the *observation&measurement* module, and *sensor&sensing* module. The final story is about the time and space conditions of phenomena or observations. Knowing the importance of these two contexts, we identify the *temporal module* that includes temporal entities and the *geospatial module* that covers spatial entities (e.g., city, country, etc.) and environmental features (such as forests). Since one of the MEMOn requirements is the reuse of existing ontologies and following the style of existing geospatial ontologies, the module geospatial ontology results from a combination of the first and the last stories.

The relationships between modules will be mapped into the ontology. Using axioms, we can, for example, ensure that environmental processes are associated with a spatiotemporal context through *Time_R* and *Spatial_R* relations. *Time_R* represents timing relations (e.g., `occurs_on`, `occurs_before`, `occurs_after`, etc.) of different disastrous weather/climate events and between them. *Spatial_R* represents spatial relations (e.g., location, neighborhood, etc.) of occurred events. The disasters also can be related to the environmental processes through causal relationships.

3.3.3 Module development phase

In this phase, the proposed ontological modules are developed and released through eight iterations and increments. The iterative process starts with the definition of specific CQs that the chosen module should answer. These CQs are discussed to define a semi-formal model which is given through a concept map created using the IHMC CmapTools computer program [1] (Figures 3.6 to 3.13). Each semi-formal model includes concepts and relationships related to these CQs by referring to existing ontologies and standards. Later, the formal model is developed using OWL2. Finally, the module is evaluated.

It is essential to define a strategy in developing ontological modules by adopting a top-down or bottom-up approach. The first one uses the top-level ontology as the basis for deriving concepts in the module. In this approach, we take advantage of the knowledge already expressed in the top-level ontology. The second one is based on identifying classes to build the ontological module, then mapping the resulted module to a top-level one. In this approach, some inconsistencies may exist between the domain ontology and the upper-level one (Noy et al., 2001). Accordingly, we use a hybrid approach based on a top-down alignment to the upper-level ontology (BFO) and the mid-level ontologies, the Common Core Ontologies (CCO) (Schoening et al., 2015), then, a bottom-up focus on classes that are grounded in environmental data sources and existing domain ontologies. These latter represent specialized domains having classes that are either direct subclasses of the upper or the mid-level ontologies (such as ENVO) or imported and classified according to BFO and CCO (such as the SSN

[1] https://cmap.ihmc.us/

ontology). In what follows, we present BFO, CCO, and MEMOn modules.

3.3.3.1 Basic Formal Ontology (BFO)

BFO (Basic Formal Ontology) is an upper-level ontology that has been applied in various domains such as biomedicine and military. It has recently become an ISO standard (ISO 21838-2). BFO provides neutral and general classes under which the domain's universals can be inserted (Arp et al., 2015). As a top-level ontology, it assists in organizing data from different domain ontologies in a way that promotes the semantic interoperability degree of systems using these ontologies. We chose BFO as a starting point for our ontology building for two primary reasons. Firstly, our domain of interest, the environment field, is realistic. Thus, we looked up for an upper-level realist ontology that represents environmental entities as they are and not representing environmental concepts and representations existing in the domain experts' minds (Smith and Ceusters, 2010). More accurately, BFO is a realist ontology. Secondly, ENVO that will be reused in our ontology is one of those ontologies that have been developed by adopting BFO as the foundation of the ontology development. These factors make its use a key enabler in promoting semantic interoperability of these ontologies and the reuse of our ontology by others. The two main categories of BFO classes are "Continuant" which are entities that continue to exist over time, such as objects and locations (forests, water, etc.) and "Occurrent," which are event entities, processes and temporal regions such as rainfall, year, etc. Figure 3.3 and Table 3.2 summarize the key categories encountered in BFO.

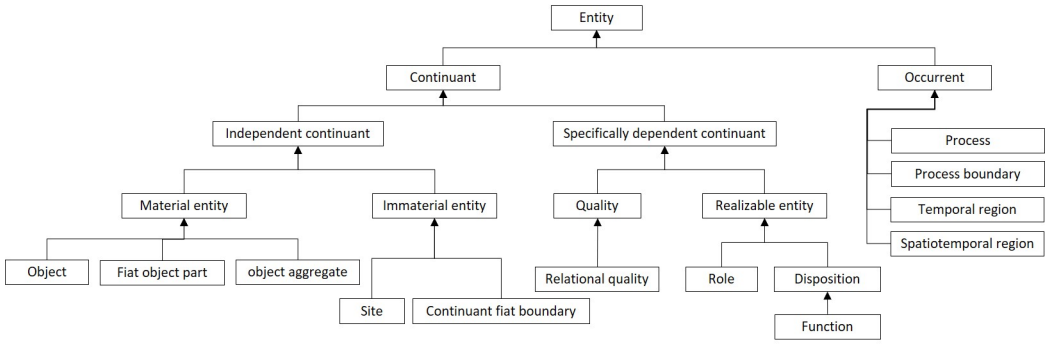


Figure 3.3: A fragment of the BFO's class hierarchy.

3.3.3.2 The Common Core Ontologies (CCO)

The Common Core Ontologies (CCO) form a collection of mid-level ontologies in OWL that extend BFO (Schoening et al., 2015). It consists of ten interrelated modules, as illustrated in Figure 3.4, allowing the development of interoperable ontologies. The ten ontologies provide a starting set of general, commonly used terms involving agents, actions, artifacts, and measurements. The ontologies in the CCO include:

Table 3.2: BFO classes definitions.

Class	Definition
“entity”	Anything that exists or has existed or will exist.
“Continuant”	An entity” that continues or persists through time while maintaining their identity and have no temporal parts. It is a dependent or independent object.
“Occurrent”	An entity that occurs happens or develops in time: events or processes or happenings.
“independent continuant”	A continuant entity that is the bearer of some qualities, it can maintain their identity and existence through gain and loss of parts, dispositions or roles, and changes in their qualities.
“generically dependent continuant”	An entity that is dependent on one or more other independent continuants. This latter can serve as its bearer. It is similar to complex continuant patterns of the sort created by authors or through the process of evolution.
“specifically dependent continuant”	An entity that depends on one or more specific independent continuants for its existence. It exhibits existential dependence and has two subcategories: quality and realizable entity.
“process”	An occurrent entity that exists in time by occurring or happening has temporal parts, and always depends on at least one material entity. It can be partitioned into temporal parts in different ways and at different levels of granularity.
“quality”	A specifically dependent continuant that depends or inheres in an entity at all and is fully exhibited or manifested or realized in that entity.
“disposition”	b is a disposition means: b is (1) a realizable entity, (2) b’s bearer is some material entity, (3) b is such that if it ceases to exist, then its bearer is physically changed, and (4) b’s realization occurs when and because this bearer is in some special physical circumstances, and this realization happens in virtue of the bearer’s physical make-up.
“function”	A disposition that exists through intentional design to realize processes.
“temporal region”	An occurrent entity that is part of the time as defined relative to some reference frame.

- Agent Ontology, representing agents, especially persons and organizations, and their roles.
- Artifact Ontology (AO), representing deliberately created material entities along with their models, specifications, and functions.
- Currency Unit Ontology, representing currencies in different countries.
- Extended Relation Ontology (ERO), representing relations (i.e., object properties) holding between entities. Extended Relation Ontology is imported in MEMOn since it contains several object properties such as "has quality," "occurs at," etc.
- Event Ontology (EO), representing processes.
- Geospatial Ontology (GeO), representing sites, spatial regions, and other entities, especially those that are located near the surface of Earth, as well as the relations that hold between them.
- Information Entity Ontology (IEO), representing generic types of information as well as the relationships between information and other entities.
- Quality Ontology (QO), representing a range of attributes of entities, including qualities, realizable entities such as dispositions and roles, and process profiles.
- Time Ontology (TO) representing temporal regions and the relations that hold between them. A temporal region, as defined by BFO, is an occurrent entity that is part of the time as defined relative to some reference frame.
- Units of Measure Ontology (UMO), representing standard units used when measuring various attributes of entities.

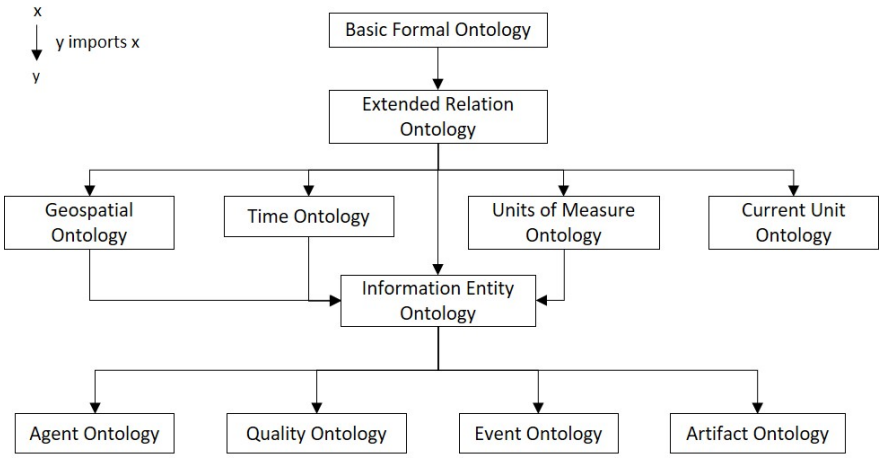


Figure 3.4: CCO modules hierarchy.

A simplified explanation of the diverse modules is presented in (Schoening et al., 2015). The CCO meets most of our requirements since it defines a modular set of extensible classes and

relations that can be used in our domain ontology. Accordingly, these mid-level ontologies[2] are extended for the development of MEMOn modules.

3.3.3.3 Semi-formal MEMOn modules building

MEMOn consists of a set of modules covering the disciplines of environmental monitoring domain. It contains eight main modules, namely:

- observation&measurement module,
- sensor&sensing module,
- natural disaster module,
- environmental process module,
- environmental material module,
- infrastructure module,
- temporal module,
- and geospatial module.

Figure 3.5 illustrates the proposed MEMOn modules and their import and reuse structure. In addition to extending BFO and CCO to define MEMOn modules, we reused existing domain ontologies, including SSN and ENVO. In addition, CCO reuse the Relations Ontology (RO), which is a collection of OWL2 relations intended to be shared among various ontologies (Smith et al., 2005). It incorporates a set of upper-level relations such as "part of" and "has input." These relations are reused in MEMOn modules.

In this section, we present the eight modules of MEMOn. For each module, we describe some classes and relations that we reused or added. We also explain how are they are inserted under BFO and CCO classes. Figures (3.6 – 3.13) illustrate partial views of the modules. In each figure, classes are marked with a color and a prefix. The prefix marks the source ontology and the namespace used to define the class. Classes added in MEMOn are marked with a namespace prefix "memon." Classes reused from existing ontologies are marked with a different namespace prefix such as "envo," "bfo," etc.,. and different colors to make the concept map easier to read (see colors in Figure 3.5). At the beginning of a module development phase, a list of specific CQs, that the module should be able to respond, is presented. The CQs shown in Tables (3.3 – 3.10) can be specialized with any given observation or event.

[2]Except agent ontology and currency unit ontology

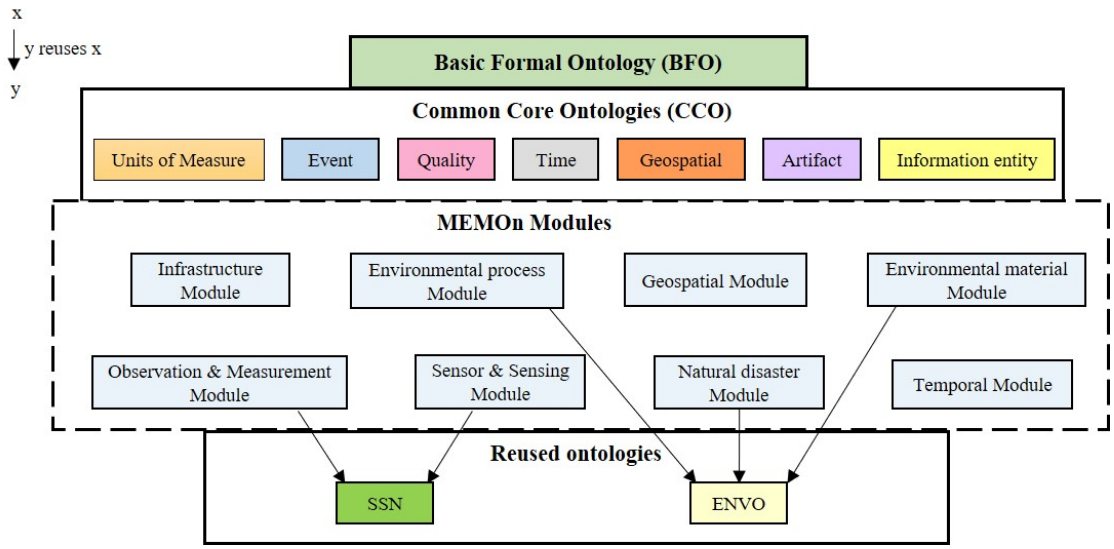


Figure 3.5: Import and reuse of ontologies in MEMOn modules.

a) Sensor and sensing module

The sensor&sensing module describes sensor systems, sensors, and how they are related to observations. This module should be able to answer the CQs defined in Table 3.3. To

Table 3.3: CQs related to the sensor&sensing module.

CQi	Competency question
1	What are the sensor categories?
2	Is sensor [X] deployed in the platform [Y]?
3	What is the sensor responsible for capturing information about the observation [X]?
4	Is sensor [X] an active or passive sensor?

develop the sensor&sensing module, we refer to SSN ontology that describes sensors and sensor networks, which is built upon the OGC standards (the SOS and the O&M standards). Besides, we reused the Artifact Ontology (AO) from CCO, which allows the classification of sensor systems as an artifact object and the definition of functions that can be applied to an artifact.

We started by importing SSN ontology. Then, we classify its classes under BFO and CCO classes. For example, the class “ssn: System” and its subclasses (Actuator, Sample, and Sensor) are classified under “AO:Artifact” since they can be defined as “An Object that was designed by some Agent to realize a certain function.” Considering that the class “Sensor” also exists in the CCO as “A Transducer that is designed to detect events or changes in its environment, and then provide a corresponding output.”, and to avoid inconsistency of the

ontology, we add an equivalent relation between “ssn: Sensor” and “AO: Sensor."

Following the classification of SSN, we define different kinds of sensors as classes in MEMOn. Such as “memon:temperature sensor," “memon:sea surface topography sensor," and “memon:pressure sensor." We also classified them into “active sensors” which are radar instruments used for measuring signals transmitted by the sensor that were reflected, refracted or scattered by the Earth’s surface or its atmosphere and “passive sensors” which are microwave instrument designed to receive and to measure natural emissions produced by constituents of the Earth’s surface and its atmosphere.

The Information Entity Ontology (IEO) of CCO defines entities related to the meaning of information artifacts. It classifies three kinds of Information Content Entity: Descriptive Entity that consists of a set of propositions that describe some entity, Designative Entity that consists of a set of symbols that denote some entity and Directive Entity that consists of a set of propositions that prescribe some entity.

The class “ssn: Output” is classified under the class “IEO: Descriptive Information Content Entity," and we added as subclasses "memon:satellite output" and "memon:sensor output." The “ssn: Deployment” is defined as “the Deployment of one or more Systems for a particular purpose.”. The definitions in BFO represent what is general in reality. So, we defined a deployment as “a methodical procedure of introducing an activity, process, program, or system to all applicable areas of an organization” and we inserted it as a subclass of “IEO: Directive Information Content Entity.”

A “Procedure” is defined in SSN as “A workflow, protocol, plan, algorithm, or computational method specifying how to make an Observation” that prescribes the inputs and outputs of a system. We classify this class under the class “AO:Artifact Function Specification” since this latter defines each Directive Information Content Entity that prescribes some artifact function. This module includes 83 classes with 1264 axioms, two data properties, and 38 object properties (see Figure 3.6).

b) Observation and measurement module

MEMOn is concerned with describing how observations are related to sensors, events, and measurements. Thus, the Observation&measurement module should be able to answer the CQs presented in Table 3.4. The goal of observation might be to measure or otherwise determine the value of a property. As the precedent module, we import SSN ontology, which defines observation, a feature of interest, and properties concepts. Then, we classify its classes under BFO and CCO classes by following the philosophy of BFO.

Observation is defined in SSN as "An act of observing a property or a phenomenon to estimate or calculate a value of a property of a feature of interest.” We classify the "sosa: observation” class under the class “EO: Act” of CCO. SOSA[3] recognizes “sosa: Observation”

[3]SOSA is a module in the SSN ontology, its prefix is sosa. https://www.w3.org/TR/vocab-ssn/

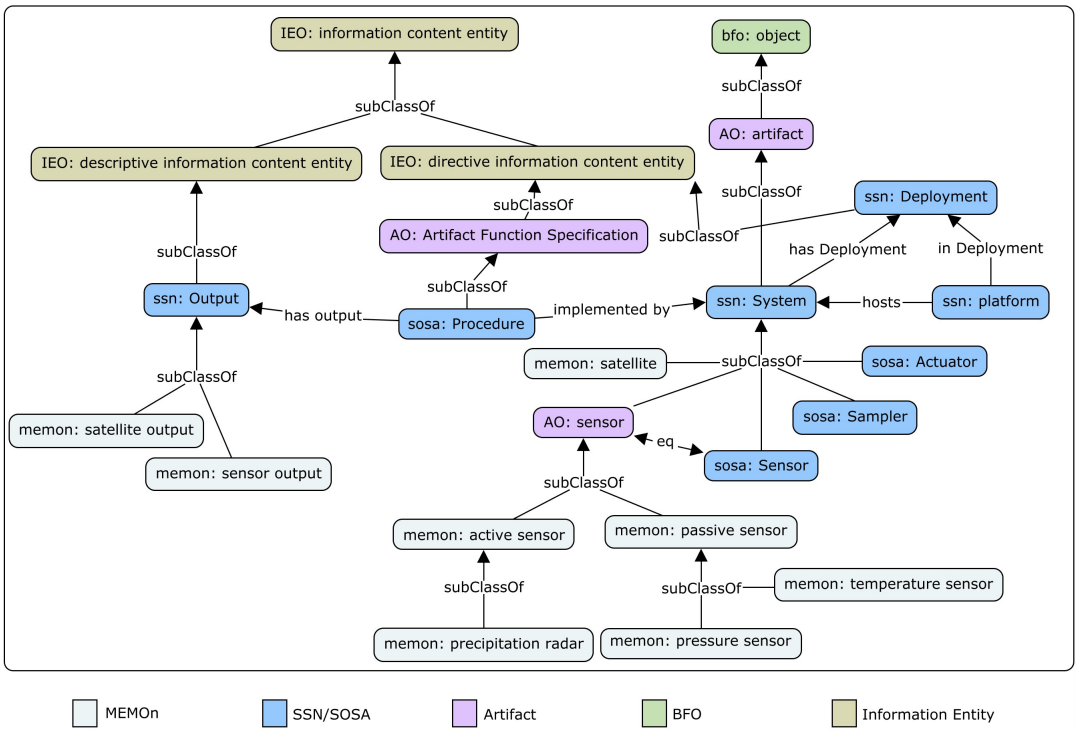


Figure 3.6: Partial view of the sensor&sensing module.

about some “sosa: ObservableProperty” of a “sosa: FeatureOfInterest”. Since the class “sosa: observableProperty”, subclass of the “ssn: Property” class, is defined as “An observable quality (property, characteristic) of a Feature Of Interest”, then it is classified under the “bfo: quality” class of BFO, whereas the class “sosa: FeatureOfInterest” is classified under “IEO: Descriptive Information Content Entity” since it represents “The thing whose property is being estimated or calculated in the course of an Observation”.

Following the classification of SSN, we define different kinds of observable properties as classes in MEMOn. To this end, we reused classes from the Quality Ontology (QA) from CCO, such as “QA: Temperature” and “QA: Wetness,” classes from ENVO such as “envo: Temperature of water” and “envo: Temperature of air.” Then, we created new classes such as “memon:precipitation,” “memon: wind speed,” “memon: humidity”, and “memon: sea surface temperature.” Figure 3.7 shows some examples of observable properties.

To reduce the generality of the class “sosa: Result,” we added the class "Observation Result." This new class is represented as the equivalent of the class “Measurement Information Content Entity” of CCO, defined as “A Descriptive Information Content Entity that consists of a symbol that is a measure of the extent, dimensions, quantity, or quality of an Entity relative to some standard.”

SSN, however, does not include modeling aspects for units of measurement and domain knowledge that are related to observed properties. To model measurement units, we reused UMO from CCO. This ontology extends from IEO the class “IEO: Measurement Unit” and

Table 3.4: CQs related to the observation & measurement module.

CQi	Competency question
1	What is the value of a given observation [X]?
2	What are the measurements for an observed property [X]?
3	What is the average measurement for a climate parameter [X]?
4	What are the measurement units used for a specific property [X]?
5	What are the observations associated with an event [X]?
6	What is the feature of interest in an observation [X]?
7	What are the observable properties?

provides several subtypes of the measurement unit, e.g., "UMO: Measurement Unit of Speed," "UMO: Measurement Unit of Pressure," and "UMO: Measurement Unit of Temperature." It also provides instances of these classes, e.g., "Horsepower Measurement Unit," and "Kilometer Measurement Unit." We consider that each observation result has a measurement unit, by adding the relation "memon: has measurement unit". Figure 3.7 shows a partial view of the observation&measurement module. This latter contains 102 classes with 968 axioms, 2 data properties, and 37 object properties.

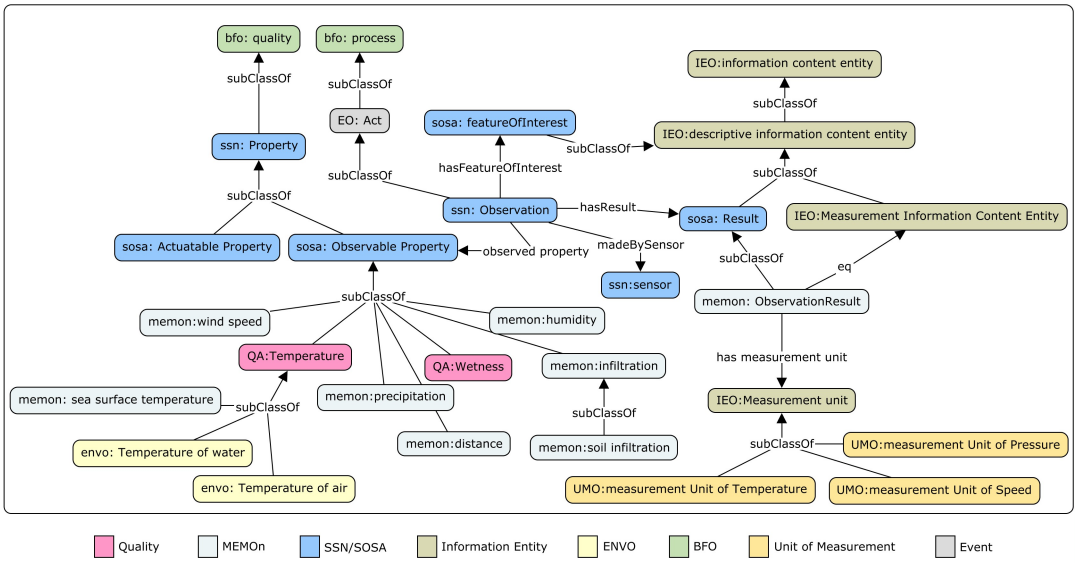


Figure 3.7: Partial view of the observation&measurement module.

c) Environmental material module

The states of environmental material play an important role in environmental monitoring. The relation between observations and environmental materials is evident. Environmental material quality (such as water quality), or environmental material existence (such as fume) are observations detected by environmental monitoring techniques and enables to monitor the environment and predict disastrous events. Before identifying the taxonomy, we define the CQs, which this module should answer (Table 3.5).

We reused classes defined under 'envo: environmental material' class in ENVO, including "envo:water", "envo:air" and "envo: rock". Figure 3.8 shows a partial view of the environmental material module. This module contains 87 classes, 802 axioms, and 6 object properties.

In OWL, classes can be either primitive or defined. Primitive classes only have necessary conditions, i.e., superclasses. Defined classes have necessary and sufficient conditions, i.e., equivalent classes. To enrich the environmental material module, we used defined classes. For instance, "memon: contaminated air" is an example of an equivalent class that satisfies necessary and sufficient conditions. In fact, we defined "memon: contaminated air" as being equivalent to "envo:air" and has quality "memon: contaminated". Similarly, 'envo: liquid environmental material' is defined as being equivalent to "envo:environmental material and has quality some "envo:quality of liquid". In other terms, if "liquid environmental material" class is described using only necessary conditions, then we can say that if an individual is a member of class "liquid environmental material" it must satisfy the conditions. We cannot say that any (random) individual that satisfies these conditions must be a member of the class "liquid environmental material." However, if the "liquid environmental material" class is now defined using necessary and sufficient conditions (our case), we can say that if an individual is a member of the class "liquid environmental material." Then it must satisfy the conditions. Also, we can say that if any (random) individual meets these conditions, then it must be a member of the class "liquid environmental material." The conditions are not only necessary for membership of "liquid environmental material" but also sufficient to determine that something satisfying these conditions is a member of "liquid environmental material." Consequently, we can use the reasoner to compute automatically a classification hierarchy in this way to classification to ensure the refinement of the individuals' classification and allow more semantic precision in the representation of knowledge in order to make the implicit knowledge more explicit. Equivalent classes are also defined in other MEMOn modules.

Table 3.5: CQs related to the environmental material module.

CQi	Competency question
1	What are the environmental materials that participate in an environmental process [X]?
2	What is the quality of an environmental material [X]?
3	Is there any relationship between an environmental material [X] and a process [Y]?
4	Is an environmental material [X] part of a geographic feature [Y]?

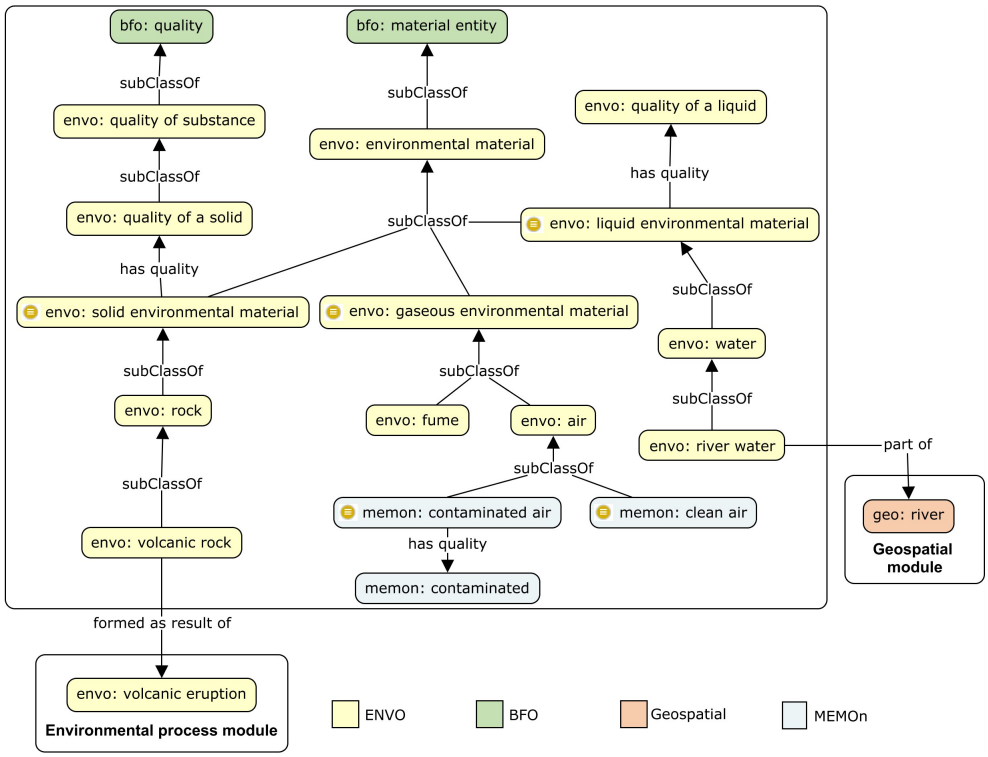


Figure 3.8: Partial view of the environmental material module.

d) Environmental process module

The understanding of environmental phenomena and environmental changes needs a complete and comprehensive representation of the processes which are involved in such changes. Consequently, a set of CQs, presented in Table 3.6, has been defined.

The environmental process module contains 214 classes representing environmental processes, and 15 relations have been added to MEMOn. The classes were aligned with the "bfo: process" class in BFO since they represent "entities that occur and develop in time, unfold in successive phases, and always depend on some material entity." They were based on the knowledge of experts, ENVO ontology, and the UNESCO thesaurus[4] and have been used to interlink disasters, environmental features, and environmental materials. For example, a "tree burning" is related to "smoke" through "has output" relation (Figure 3.9). The relations between the classes of the environmental process module and other MEMOn modules are primarily controlled by RO such as "ro: has input" and "ro: has output". Also, we added new relationships such "memon: occurs during" and "memon: has participant".

Following the classification of ENVO, we define different kinds of environmental processes and classify them into climatological, hydrological, geographical, geophysical, and other categories. In addition to the different main categories of environmental processes,

[4]http://vocabularies.unesco.org/browser/thesaurus/en/

Table 3.6: CQs related to the environmental process module.

CQi	Competency question
1	What are the categories of environmental processes?
2	What are the environmental processes that caused the environmental process [X]?
3	Is there any relationship between environmental processes [X] and [Y]?
4	What are the subtypes of the environmental process [X]?
5	What are the environmental processes preceding an environmental process [X]?
6	What are the environmental processes resulting from an environmental process [X]?
7	Is environmental process [X] follows environmental process [Y]?
8	What are the environmental processes that caused the disaster [X]?

we reused classes from Event Ontology (EO) of CCO to describe changes such as the classes "eo:Decrease of a quality" and "eo:Loss of a quality" and we extended them with classes such as "memon:Decrease of a temperature", "memon: Loss of a mass".

In the aim of better understanding environmental phenomena, their factors, and especially their order of occurrence, we identified the need to add object properties such as "memon: caused by" and "ro: preceded by."

e) Natural disaster module

Although the strong relationships between environmental processes and disasters, the definition of natural disasters was set aside in a separate module, this separation will promote its reuse for other applications such as an emergency response application. CQs defined for the natural disaster module were summarized in Table 3.7.

Table 3.7: CQs related to the natural disaster module.

CQi	Competency question
1	What are the different types of natural disasters?
2	Which natural disasters may lead to natural disaster [X]?
3	Is there any relationship between natural disaster [X] and natural disaster [Y]?
4	Is the natural disaster [X] is followed by a natural disaster [Y]?

In ENVO, a disaster is modeled as a subcategory of the class "bfo: process." However, to emphasize the difference between natural disasters and environmental processes, we classify

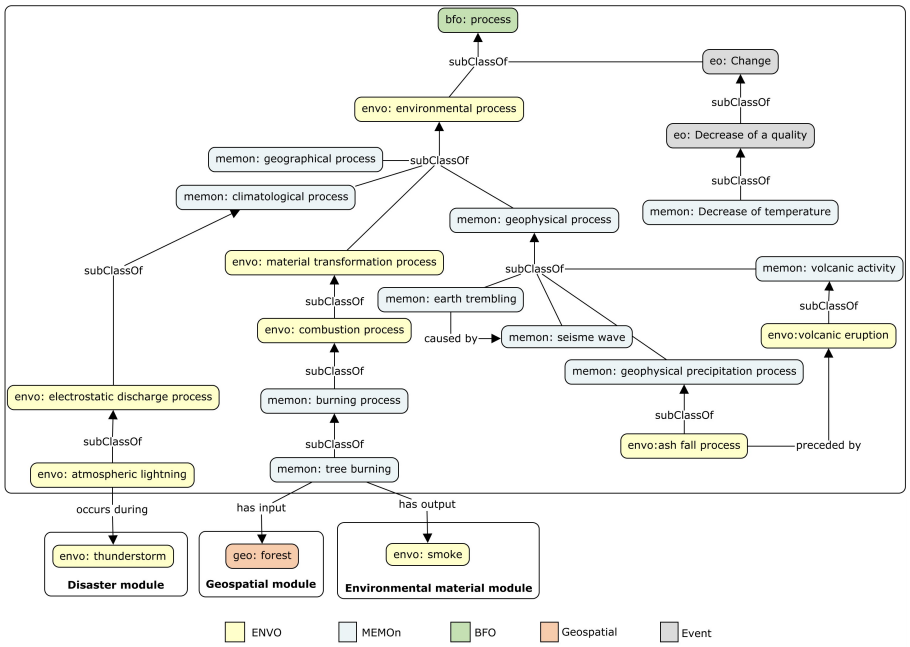


Figure 3.9: Partial view of the environmental process module.

natural disasters as "bfo: disposition." According to BFO, a disposition is "a realizable entity in virtue of which a process of a certain kind occurs in the independent continuant (bearer) in which the disposition inheres. This process is called the realization of the disposition", for example, the disposition of a forest region –which undergoes land degradation – to desertification. Thus, we classified the desertification disaster under the class "bfo: disposition" and specified the class "land degradation" as a process.

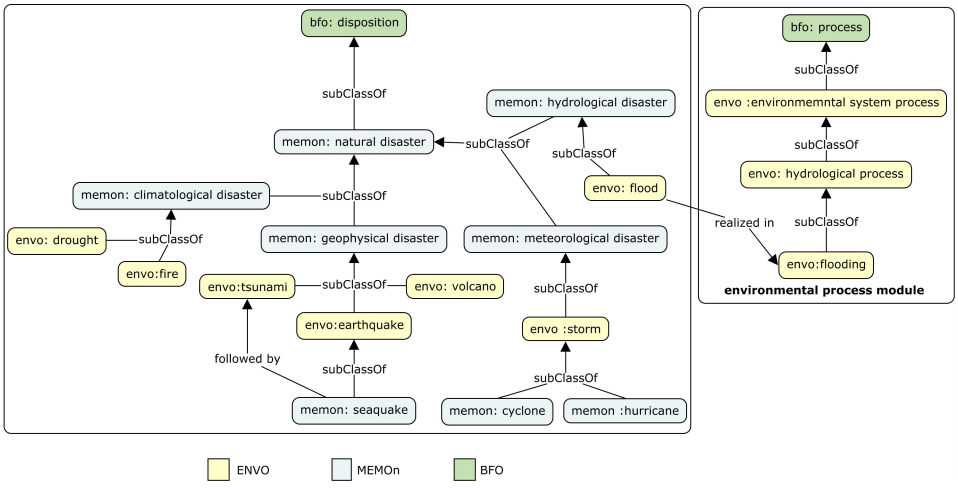


Figure 3.10: Partial view of the natural disaster module.

We define different kinds of natural disasters in the MEMOn ontology and classify them into climatological, geophysical, hydrological, and meteorological categories. We reuse classes

from ENVO such as “envo: earthquake” and “envo: volcano” and create new classes such as “memon: cyclone” and “memon: hurricane.” In addition to classifying natural disasters into different categories, we also define them regarding how they affect each other by establishing the relationships “ro:caused by,” and “memon:followed by” and how they occurred as a result of one or many environmental processes by the relation “bfo:realized in," as illustrated in Figure 3.10. Figure 3.10 shows a partial view of natural disaster module classes. The entire module contains 80 classes, 782 axioms, and 4 relationships.

e) Infrastructure module

Several physical objects mentioned in MEMOn could be useful in environmental monitoring. Infrastructure objects which are deployed in a specific location where particular environmental processes occur could participate in environmental hazard factors, for example, the dam failure flood in Laos. We defined CQs related to the infrastructure story in Table 3.8.

Table 3.8: CQs related to the infrastructure module.

CQi	Competency question
1	What are the types of infrastructure?
2	Is the infrastructure [X] participating in the occurrence of the process [Y]?
3	What is the impact of a natural disaster [X] on hydraulic infrastructure?

To build this module, we import classes from AO ontology from CCO. AO includes transportation infrastructure such as bridges and tunnel. For better classification, we added the class “memon: hydraulic infrastructure” to incorporate water infrastructures such as “AO: dam”, “memon:valve”, and “memon:sewer” as shown in Figure 3.11. The infrastructure module includes 48 classes and 608 axioms.

g) Geospatial module

The challenges in environmental monitoring are motivated by the necessity to take into consideration geospatial information. According to Beard and Neville, “Knowledge of spatial contextual differences among observations is important for interpretation and analyses” (Beard and Neville, 2014). We addressed the spatial aspect after defining the CQs related to the geospatial discipline, as illustrated in Table 3.9.

The identification of the CQs core elements (classes and their relationships) enables the creation of the module vocabulary that consists of defining environmental features, locations, and relationships among them by importing the gesospatial ontology (GEO) of CCO. An

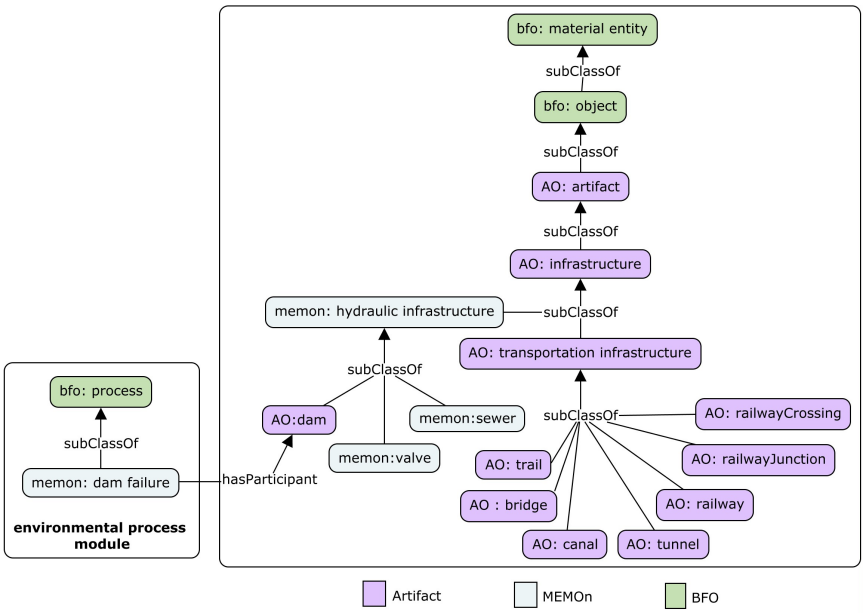


Figure 3.11: Partial view of the infrastructure module.

Table 3.9: CQs related to the geospatial module.

CQi	Competency question
1	Where did the natural disaster [X] take place?
2	Where an observation [X] was observed?
3	What are the geographic coordinates of a country [X]?
4	What are the cities of a country [X]?
5	To which continent does the country [X] belong?

“environmental feature” as defined by GEO is “a site/feature having a (relatively) stable location in some geospatial region which can be designated by location-specific data." It contains four categories: anthropogenic features, which are features relating to or resulting from the influence of human beings on nature, geographic features that are natural features, marine features which have characteristics of a marine entity and finally habitat which are environment features having qualities which may sustain an organism or a community of organisms.

Referring to the W3C Basic Geo (WGS84 lat/long) vocabulary , we represent longitude and latitude and other information about spatially-located things. Specifically, we add the class “memon: geographical coordinate” and two datatype properties; “has latitude value” and “has longitude value". Indeed, we distinguish two primary types of properties in this module: object properties that link classes to other classes and datatype properties that link classes to data values. As an example of object properties, we can cite “memon: located in"

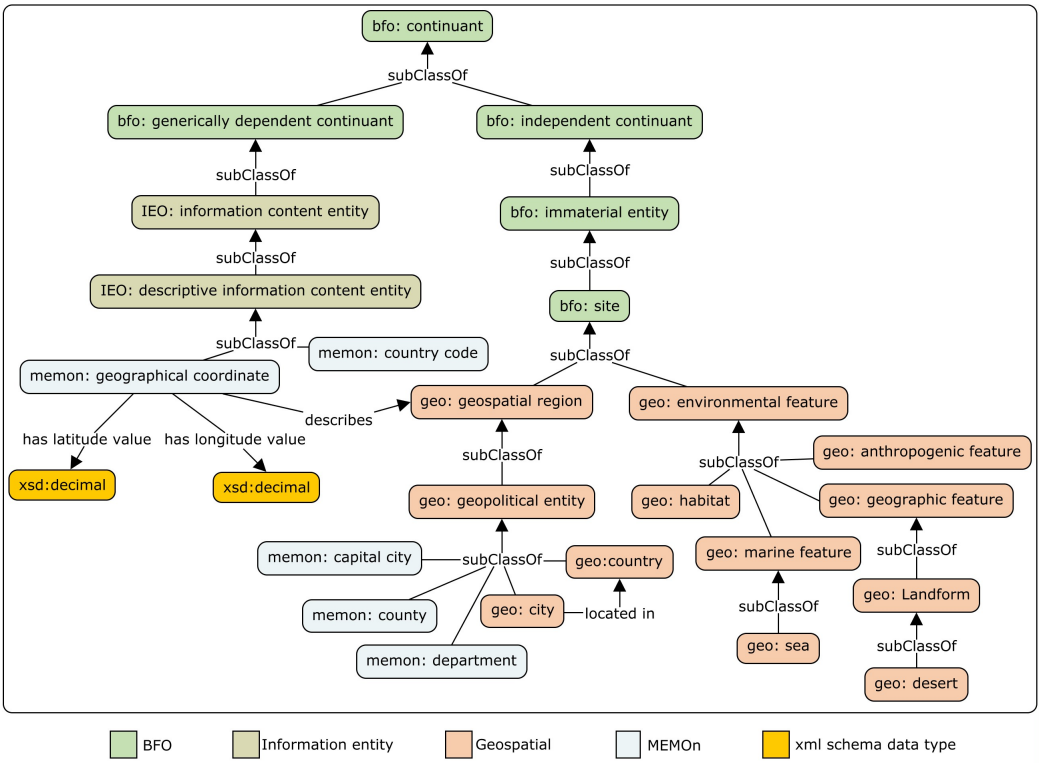


Figure 3.12: Partial view of the geospatial module.

that links a “geo:city” to a “geo:country”. Figure 3.12 presents a partial view of the MEMOn geospatial module. This module includes 291 classes, 2216 axioms, 21 object properties and 2 data properties.

h) Temporal module

In addition to the spatial context, the temporal context of observations and events is also essential to understand and monitor environmental phenomena. The CQs related to the temporal module are summarized in Table 3.10. Several ontologies have been developed to describe temporal information — for example, SWRL Temporal Ontology and the W3C’s Time Ontology . To model temporal information, we choose the Time Ontology (TO) from CCO, while it is based on BFO and interoperable with other modules. Additionally, it offers a set of object properties (e.g. "TO: interval during," “TO: instant is before”) that can be used to reason with temporal information.

Since the temporal information from different sources is heterogeneous, we add some defined classes to represent the different representations of the temporal information. For example, the class “memon:date”, the class “memon:timestamp” that is equivalent to a “memon:date” and contains a “TO:Day”, a “TO:Month”, a “TO:Year”, an “TO:Hour”, a “TO:Minute”, and a “TO:Second” and the class “memon:year-month” that is equivalent to

a “TO:Year” and a “TO:Month” are added to the module. Figure 3.13 presents a partial view of the temporal module classes and properties. We added and asserted some object properties as sub-properties of corresponding properties in the TO, such as “memon: day” which is asserted as a sub-property of “TO: interval contains”. “memon:`occurs_before`” and “memon:`occurs_after`” are other examples of object properties added to the ontology to describe timing relations between different environmental processes. The entire module contains 60 classes, 1184 axioms, and 60 object properties.

Table 3.10: CQs related to the temporal module.

CQi	Competency question
1	When did an event [X] take place?
2	What is the date [X] when an observation value [Y] is the highest?
3	Does the process [X] occur before the process [Y]?
4	Does the process [X] occur at the same time as the process [Y]?

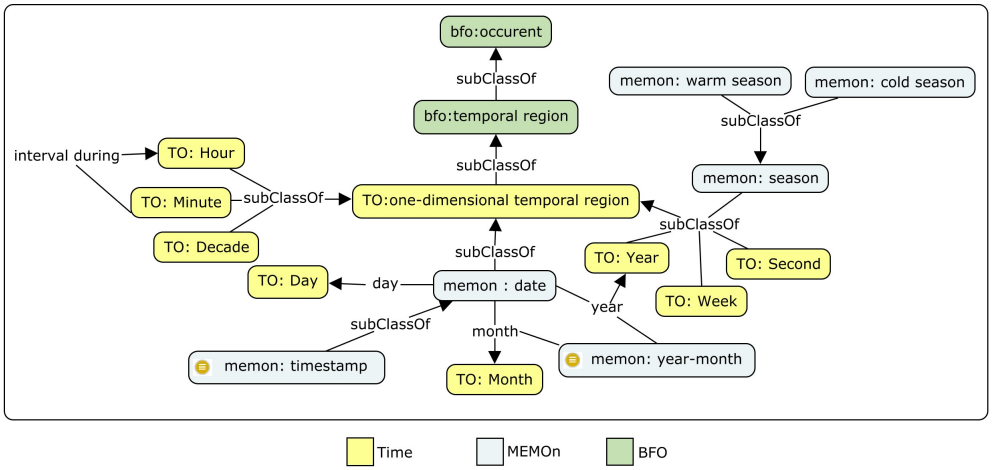


Figure 3.13: Partial view of the temporal module.

3.3.3.4 Module formalization and evaluation

The building process of MEMOn modules is supported by the use of Protégé (Protégé, 2014), a free, popular, and open-source platform that provides users with a set of tools to construct semantic models. It is an OWL ontology development environment. It greatly facilitates the definition of classes, properties, restrictions, and rules of MEMOn and supports the visualization and manipulation of the ontology. MEMOn is expressed in OWL2. For the reuse of existing ontologies such as CCO, we used the “owl: import” feature of OWL2. Indeed, the import of a specific class from an ontology A to an ontology B is equivalent to a copy of all the declarations of this class, including URIs, labels, definitions, and relations.

After the formalization step, the developed module is evaluated. Firstly, the consistency of the module is checked through the reasoner Pellet, which is an OWL2 reasoner, integrated into the Protégé software. Indeed, ontology inconsistency refers to the fact that there are logical incoherencies or modeling problems that arise if the primitives given by the ontology implementation language are not used correctly (such as contradictory relations). Secondly, the module is evaluated in comparison with specific criteria. Various approaches have been proposed in the literature for ontology evaluation, targeting several criteria and different metrics (Gangemi, 2005), (Obrst et al., 2007). We used the Tool for Ontology Modularity Metrics (TOMM) software, which encompasses all the evaluation metrics presented in (Khan and Keet, 2016). These metrics are grouped into four categories of criteria:

- The structural criteria are calculated based on the structural and hierarchical properties of the module. Calculating structural criteria involves evaluating:
  - The size, i.e., the number of entities in a module (|M|) such as the number of classes, object, and data properties.
  - The relative size represents the size of the module compared to the global ontology O.
$$Relative\ size(M) = \frac{|M|}{|O|} \tag{3.1}$$
  - The Atomic size: is the average size of a group of interdependent axioms in a module. An atom in a module is defined as a group of axioms with dependencies between each other.
$$Atomic\ size(M) = \frac{|Axioms|}{|Atom|} \tag{3.2}$$
  - The Cohesion, which refers to the extent to which entities in a module are related to each other.
$$Cohesion(M) = \begin{cases} \sum_{E_i \in M} \sum_{E_j \in M} \frac{SR(e_i, e_j)}{|M|(|M|-1)} & if |M| > 1 \\ 1 & otherwise \end{cases} \tag{3.3}$$
    where SR(e_i; e_j) is the relation function.
- The logical criteria include the correctness and the completeness metrics.
  - The correctness evaluates if every axiom that exists in the module also exists in the global ontology.
$$Correctness(M) = \begin{cases} true & if Axioms(M) \subseteq Axioms(O) \\ false & otherwise \end{cases} \tag{3.4}$$
  - The completeness evaluates if the meaning of every entity in a module is preserved as in the global ontology.
$$Completeness(M) = \ true \ \ if \sum_i^M Axioms(Entity_i(M)) \models Axioms(Entity_i(O)) \tag{3.5}$$

- The relational criteria that deal with the relations and behavior that modules exhibit with each other such as:

  - The Inter-Module Distance (IMD) in a set of modules which describes the number of modules that have to be considered to relate two entities.

$$IMD(M) = \begin{cases} \sum\limits_{E_i,E_j \in (M_i,M_n)} \dfrac{NM(E_i,E_j)}{|(M_i,M_n)|(|(M_i,M_n)|-1)} & if |(M_i,M_n)| > 1 \\ 1 & otherwise \end{cases} \tag{3.6}$$

    Where NM(E_i, E_j) is the number of modules to consider to relate entities i and j, the product of |(M_i,M_n)|(|(M_i,M_n)|-1) represents the number of possible relations between entities in a set of modules M_i,M_n.

- The richness criteria include:

  - The Attribute Richness which defines the average number of attributes per class, where att is the number of data properties in the module.

$$AR(M) = \frac{|att|}{|C|} \tag{3.7}$$

  - The Inheritance Richness which is defined as the number of subclasses (|H|) per class (|C|) in a module.

$$IR(M) = \frac{|H|}{|C|} \tag{3.8}$$

  - The Relationship richness that describes the diversity of relations types in the ontology. P is the set of non-hierarchical properties.

$$RR(M) = \frac{|P|}{|H| + |P|} \tag{3.9}$$

According to the metric results applied to MEMOn, shown in Table 3.11, we can deduce the following points:

- The atomic sizes of the modules indicate that there are an average between 3.23 and 4.79 axioms that are grouped in an atom for the eight modules.

- Most of the MEMOn modules contain few attributes, as the attribute richness is less than 1 for the majority of the modules.

- $M_{Disaster}$ has the lowest RR and $M_{Temporal}$ has the highest one. An ontology that has little value of RR may have only inheritance relationships. That is the case of $M_{Disaster}$ since it includes only the set of disasters organized hierarchically. Consequently, it conveys less information than $M_{Temporal}$, which contains a diverse set of relationships (e.g., “interval during” and “has ending instant”).

- IR values are comprised between 1 and 2.9, which represent high values. Indeed, Ontological modules with high IR are called vertical ontologies since classes have a large number of direct subclasses. This indicates that our modules represent a wide range of knowledge with a low level of detail. Accordingly, they are more open to evolving and being specified. This evolution corresponds to our objective to enrich the ontology with further information in other development iterations.

- By comparing the cohesion values (coh(M)), we found that the modules $M_{Temporal}$, $M_{O\&M}$ and $M_{EnviProcess}$ have the highest cohesion values, due to the strong relatedness of different classes of each module. For instance, the observation and measurement module $M_{O\&M}$ deals with the classes of observed properties. This module models relations between observation events, measured properties, and measurement units. All of these classes and how they are related are the essence of the higher cohesion in the ontological module.

Table 3.11: MEMOn Metrics' results.

Modules	Structural metrics					Logical metrics		Relational metrics	Richness metrics		
	$N_{classes}$	$N_{properties}$	N_{axioms}	Atomic size	Coh	Correctness	Completeness	IMD	AR	IR	RR
$M_{Disaster}$	80	4	782	3.43	0.04	true	true	4920	0.2	2.2	0.021
$M_{EnviProcess}$	214	15	1083	3.27	0.067	true	true	91949	0.0469	2.873	0.122
$M_{EnviMaterial}$	87	6	802	3.543	0.032	true	true	2595	0.62	2.6	0.159
M_{Sensor}	83	40	1264	4.79	0.008	true	true	902	1.79	2.16	0.147
$M_{O\&M}$	102	39	968	3.23	0.068	true	true	1636	0.024	2.93	0.167
$M_{Geospatial}$	291	23	2216	4.615	0.008	true	true	1203	1.2	1.045	0.093
$M_{Temporal}$	60	60	1184	3.583	0.18	true	true	845	0.06	1.917	0.3125
$M_{Infrastructure}$	48	2	608	3.68	0.0164	true	true	194	0.042	2.16	0.125

In conclusion, the result metrics values generated by TOMM software represent valuable measurements to evaluate the ontology modules. The main objectives of the evaluation step via metrics are to (1) check the consistency of each module and between modules and the global ontology, and (2) provide an insight to the ontology users about the granularity of each module. If it is not highly completed (in terms of classes, relations), they should refine it with more specific classes and relationships according to their reuse context.

3.3.4 Release phase

The release phase is composed of three steps. We start by merging the developed modules to construct MEMOn. Then, we enrich the global ontology by inter-module relations, and finally, we evaluate it.

3.3.4.1 MEMOn implementation and enrichment

To implement the proposed ontology, we merged the eight developed modules together and integrated them into one ontology MEMOn. After that, the global modular ontology is enriched by inter-module relations. Figure 3.14 provides an overview of MEMOn, revealing the different modules and presenting some inter- and intra-module relationships. Each module has at least one relationship with other modules. For example, the disaster module is linked to the environmental process module by the relation “realized in," this means that a disaster can be produced when one or many environmental processes occur. The environmental process module is linked with the temporal module by the relation “occurs on” and with the geospatial module by the relationship “occurs at". These connections seek to identify the time and location of a process. Table 3.12 presents the number of classes and relations of the global ontology.

Table 3.12: Classes and relations of MEMOn

Ontology	Number of classes	Number of relations
Developed ontology		
MEMOn	558	47
Imported ontologies		
BFO	27	-
CCO	368	253
SSN	17	11
Classes from existing ontologies		
ENVO	251	-
RO	-	67
Total	1221	378

Some complex relationships cannot be represented by OWL2-DL. For example, OWL2 cannot express the relations between individuals referenced by object properties. To over-

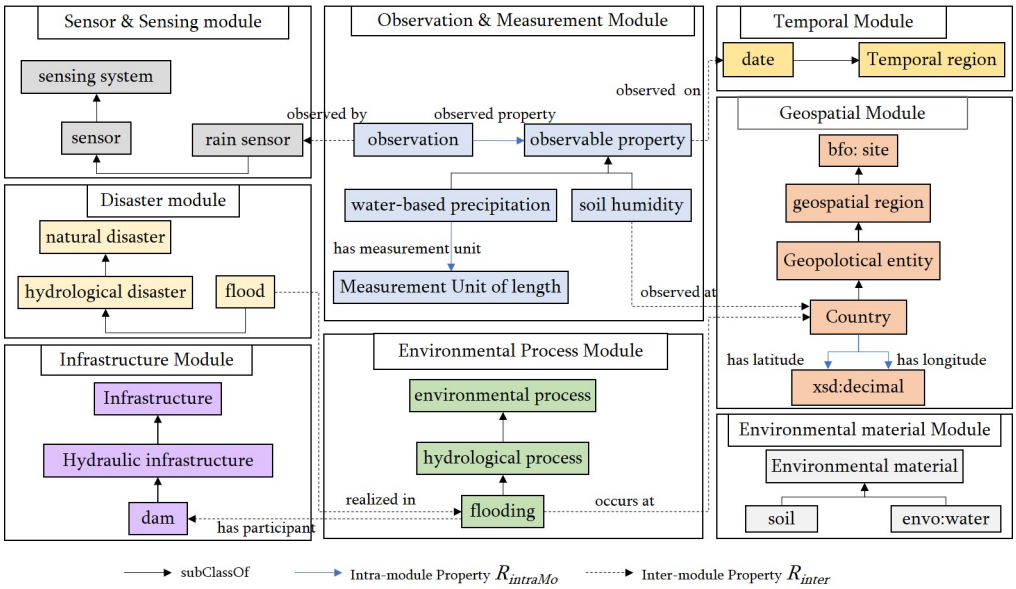


Figure 3.14: Partial view of MEMOn modules.

come the expressiveness limits of OWL2, we use SWRL language. Table 3.13 summarizes some examples of SWRL rules specified regarding the MEMOn ontological vocabulary. For instance, rule R1 automatically classifies an existing rain, with a precipitation rate between 16 and 50 mm/hour, as a "very heavy rainfall." Indeed, in our ontology, there are several categories of rain process: "light rainfall," "moderate rainfall," "heavy rainfall," and "very heavy rainfall." This classification is due to the precipitation detection rate. Other rules with the same syntax as R1 allow the classification of the other rain categories. At the beginning of a reasoning process, we input different rains as instances of the "water-based rainfall" class; we still do not know the category of each rain process. Once the reasoning process is finished, the results obtained show that each detected rain has been inserted under the corresponding category of rain. Hence, implicit knowledge is generated from inputted data.

3.3.4.2 MEMOn evaluation

In this subsection, we present the evaluation of MEMOn through the verification and validation steps.

a) MEMOn verification

Ontology verification is "the ontology evaluation which compares the ontology against the ontology specification document, thus ensuring that the ontology is built correctly" (Suarez-Figueroa and Gómez-Pérez, 2008). In other words, it allows us to answer the question, "Are we producing the ontology, right?". To do so, we first checked if MEMOn contains any inconsistency by using the Pellet reasoner. Second, to verify it against the requirements and objectives defined at the beginning of the development process, we translate some CQs to

Table 3.13: Some examples of MEMOn's SWRL rules.

Ri	SWRL rule
R1	Classify an environmental process to a specific type of process: 'water-based rainfall'(?r), precipitation(?p), 'has precipitation value'(?r, ?p), swrlb:greaterThan(?p, 16), swrlb:lessThan(?p, 50) → 'very heavy rainfall'(?r)
R2	Relate flooding process to the cause of very heavy rain: 'very heavy rainfall'(?r), date(?d), 'spatial region'(?rg), 'occurs on'(?r, ?d), 'occurs at'(?r, ?rg), flooding(?f), 'occurs on'(?f, ?d), 'occurs at'(?f, ?rg) → 'caused by'(?f, ?r)
R3	Relate a soil type to limited infiltration: soil(Regosol), "located in" (Regosol, ?rg), "spatial region"(?rg), "soil infiltration" (?inf) → "limited soil infiltration"(?inf)
R4	Reclassify a general disaster to a specific type of disaster: hurricane(?h), cyclone_process (?p),realized_in(?h,?p), atmospheric_wind(?w), has_part(?p,?w), storm_surge(?s), has_part(?p,?s), barometric_pressure(?bp), has_quality(?p,bp),wind_speed(?ws), has_quality(?w,?ws), measurement_value(?wv), has_measurement_value(?ws,?wv), swrlb :greaterThan(?wv, 252), has_unit(?ws, ?su), measurementUnitOfSpeed(?su), has_value (?su,"km/h") → hurricane_category_5(?h)

SPARQL language in order to query the ontology. Examples of obtained results are presented in the figures (3.15, 3.16, and 3.17).

- CQ1: What are the environmental processes that cause a flood disaster [`ENVO_01000710`]?
- CQ2: What are the environmental materials involved in an effusive volcanic eruption?
- CQ3: What is the type of sensor used for ground vibration [`memon_00001122`]?

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX obo: <http://purl.obolibrary.org/obo/>

SELECT ?disaster ?Prop ?environmentalProcess
WHERE {
?disaster (rdfs:subClassOf|(owl:equivalentClass/owl:intersectionOf/rdf:rest*/rdf:first)) ?x .
?disaster rdfs:subClassOf obo:ENVO_01000710 .
?x owl:onProperty ?Prop ;
  owl:someValuesFrom ?environmentalProcess .
}
```

disaster	Prop	environmentalProcess
flood	'realized in'	flooding
flood	'realized in'	'heavy rainfall'
flood	'realized in'	'dam failure process'

Figure 3.15: SPARQL query and results of the CQ1.

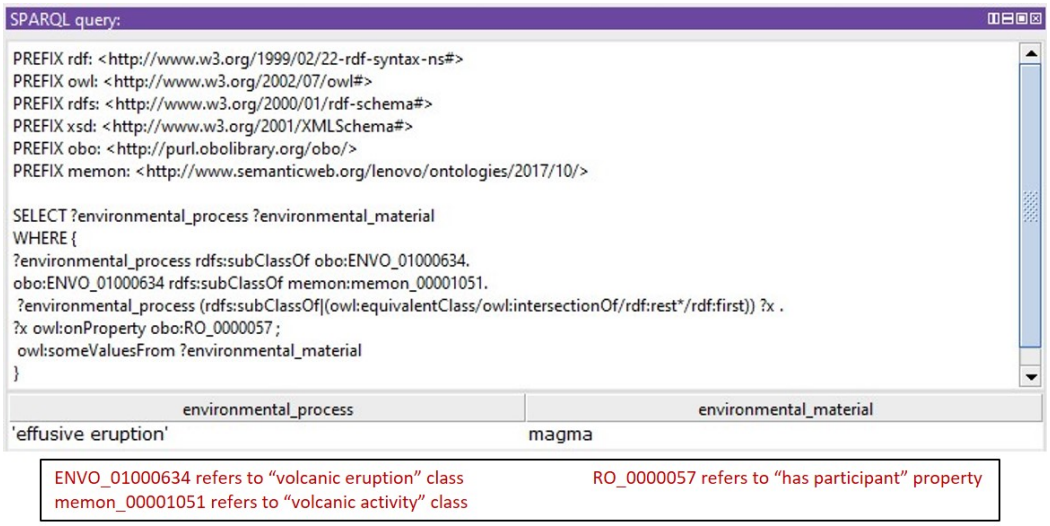


Figure 3.16: SPARQL query and results of the CQ2.

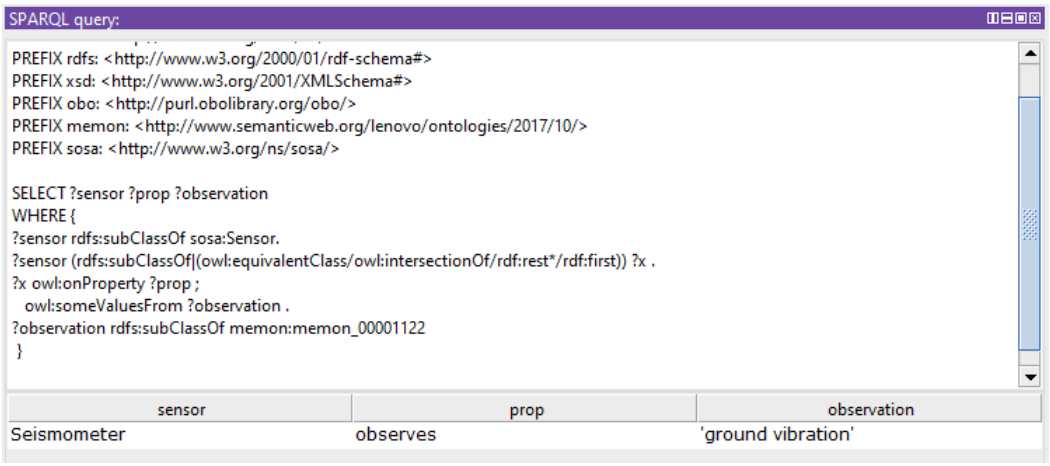


Figure 3.17: SPARQL query and results of the CQ3.

The answers to these questions, verified by the domain experts, show that MEMOn has the competency in providing the right information to the questions.

b) MEMOn validation

Ontology validation is "the ontology evaluation that compares the meaning of the ontology definitions against the intended model of the world aiming to conceptualize" (Suarez-Figueroa and Gómez-Pérez, 2008). In other words, it allows us to answer the question, "Are we producing the right ontology?". To validate MEMOn; we will focus on two case studies: (1) the exploitation and reuse of environmental knowledge, and (2) the multi-source data integration.

Knowledge exploitation and reuse:

We present, here, an example to check the possibility to explicit implicit knowledge expressed into MEMOn. To do so, we instantiate the class “hurricane” of MEMOn with the instance “IRMA” that occurred in the Caribbean in 2017. Then, we mapped this instance with characteristic instances that describe the IRMA hurricane, such as the value 914 mb for the barometric pressure, 285 km/h for the wind speed, and 5.4 m for the sea level according to the conceptual model illustrated in Figure 3.18. After that, we used the reasoner Pellet to infer implicit knowledge from explicit one. Finally, we interrogate MEMOn with the SPARQL query illustrated in Figure 3.19 to extract the list of the five-category hurricanes [`memon_00001186`], which exists in MEMOn knowledge base.

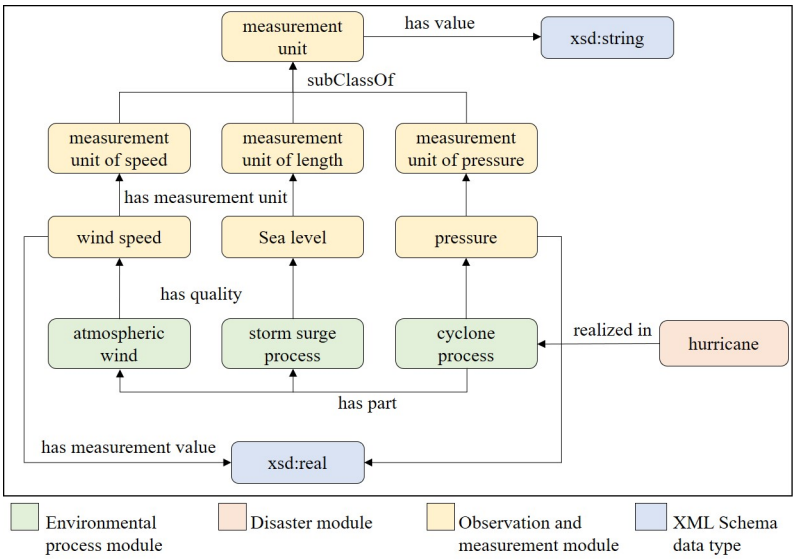


Figure 3.18: The part of MEMOn used to define a five-category hurricane. Each MEMOn module is marked with a unique color. XML schema datatypes are shown in blue.

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX memon: <http://www.semanticweb.org/lenovo/ontologies/2017/10/>
SELECT ?hurricane
        WHERE { ?hurricane rdf:type memon:memon_00001186 }
```

hurricane
IRMA

Figure 3.19: SPARQL query and the result of the 5-Category hurricane.

Thanks to the rule R4 (Table 4.14) that automatically classifies an instance of the hurricane class, knowing that it occurred with a wind speed greater than 252 km/h, as a “five-category hurricane." We get “IRMA” as a result of this query. This information was not explicitly instantiated in the ontology (was implicit), but, through the defined rule, the knowledge concerning the category of the hurricane can be inferred from the knowledge base and

becomes explicit.

Multisource data integration:

In the second use case, we present an example to demonstrate the ability of MEMOn to guarantee semantic interoperability, integrate, and link observed data from multiple sources. This use case is based on a real-world example and deals with precipitation data from the OSS in RASTER images format, data about storm events from the NOAA, and data about flood events from the EMDAT in CSV format. One of the problems that we were faced with integrating data across sources is that data schemas (metadata) in the three sources are heterogeneous. Terms are presented differently in each source. For example, in EMDAT, the term 'disaster' is used to describe a natural disaster, while the term 'event' is used in NOAA. Thus, MEMOn is used to resolve this problem and ensure semantic interoperability between heterogeneous data schemas.

To semantically integrate this multi-source data, we used the Karma Web system (Knoblock et al., 2012), a data modeling and integration framework. Karma provides tools to semi-automatically build a semantic model of a data source. This model makes it possible to rapidly map a set of sources (represented in XML, CSV, JSON, structured text files, or databases) into a domain ontology. Once the data sources are modeled, the models are then converted into a variety of formats, including RDF.

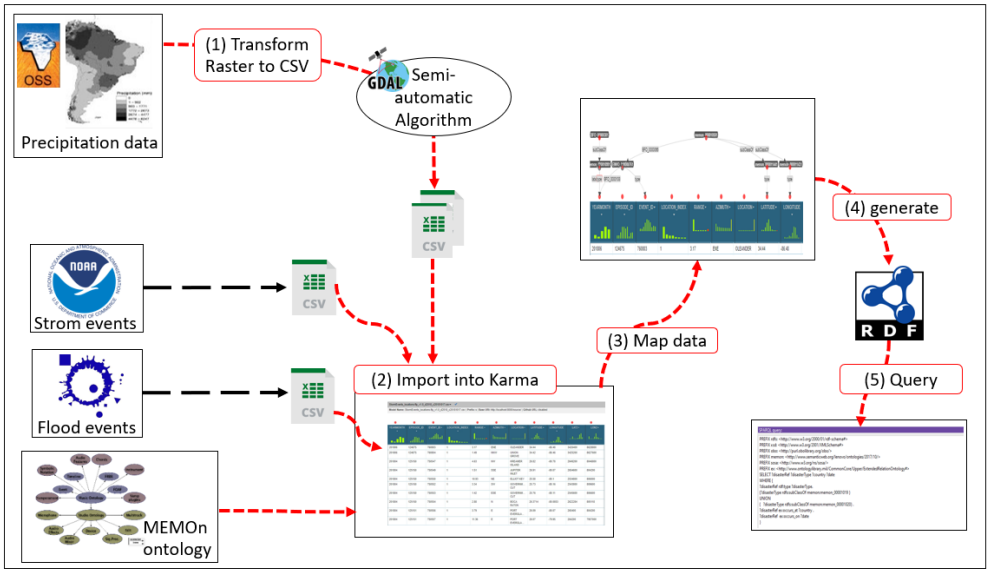


Figure 3.20: A global schema of the use case deployment.

Figure 3.20 illustrates the whole use case process. Before importing data into karma, we transformed the raster images provided from the OSS into CSV files (Step 1 in Figure 3.20). To this end, we have designed and implemented a semi-automatic process that performs this projection, using the Geospatial Data Abstraction Library (GDAL) [5]. Then, we imported data from the multi-source CSV files into karma, where MEMOn is also imported as the

[5]http://www.gdal.org/

required ontology for the data mapping process (Step 2). Figure 3.21 shows the snapshot of an example of the data importation in Karma.

Karma provides a graphical user interface to let users interactively model the data for each source according to the ontology. Modeling is the process of specifying how the different metadata items of a data source (columns' names in a CSV file) map to classes and relationships in an ontology. The data mapping involves two interleaved steps; the assignment of classes to data columns' names and the specification of the relationships, extracted from MEMOn, between the classes, as illustrated in Figure 3.22. For example, we map the metadata item "`EVENT_ID`" to the class "`ENVO_01000876`", that refers to "storm" from MEMOn and the metadata item "YEARMONTH" to the class "`memon_00001099`", labeled "YearMonth" in the ontology. Then, we link the two classes with the relation "`occurs_on`", as illustrated in Figure 3.22 (Step 3). The output of the mapping process is the semantic models corresponding to data sources (CSV files in our case). Once we have modeled the three data sources, karma uses these models to convert the data into an RDF data model that describes the global view of data (Step 4).

This RDF model is used to perform SPARQL queries across the three data sources (Step 5). We consider two examples of queries that need information for more than one source. In doing so, we used the SPARQL Query editor integrated into the Protégé software.

StormEvents_locations-ftp_v1.0_d2018_c20181017.csv

Model Name: StormEvents_locations-ftp_v1.0_d2018_c20181017.csv | **Prefix:** s | **Base URI:** http://localhost:8080/source/ | **Github URL:** disabled

YEARMONTH	EPISODE_ID	EVENT_ID	LOCATION_INDEX	RANGE	AZIMUTH	LOCATION	LATITUDE	LONGITUDE	LAT2	LON2
201806	124675	760803	1	3.17	ENE	OLEANDER	34.44	-86.48	3426400	8628800
201806	124675	760804	1	1.49	NNW	UNION GROVE	34.42	-86.46	3425200	8627600
201804	125150	750547	1	4.63	NW	KREAMER ISLAND	26.82	-80.78	2649200	8046800
201804	125150	750549	1	1.51	SSE	JUPITER INLET	26.91	-80.07	2654600	804200
201804	125150	750550	1	10.93	NE	ELLIOT KEY	25.58	-80.1	2534800	806000
201804	125150	750552	1	3.34	SW	GOVERNM... CUT	25.73	-80.16	2543800	809600
201804	125150	750553	1	1.42	ESE	GOVERNM... CUT	25.76	-80.11	2545600	806600
201804	125150	750554	1	2.88	N	BOCA RATON	26.3714	-80.0853	2622284	805118
201804	125151	750556	1	3.79	E	PORT EVERGLA...	26.09	-80.07	265400	804200
201804	125151	750557	1	11.36	E	PORT EVERGLA...	26.07	-79.95	264200	7957000

Figure 3.21: Storm events data extracted from the NOAA web service.

Following are the two queries:

1. What are the natural disasters that occurred between 2016 and 2018?
2. What are the amounts of the annual precipitation of the flood events that occurred between 2016 and 2018?

Query1: What are the natural disasters that occurred between 2016 and 2018?

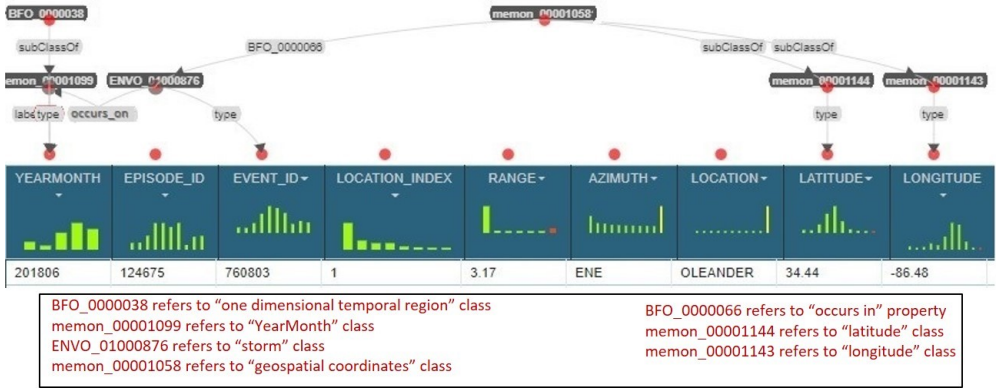


Figure 3.22: Snapshot of Karma for the data mapping.

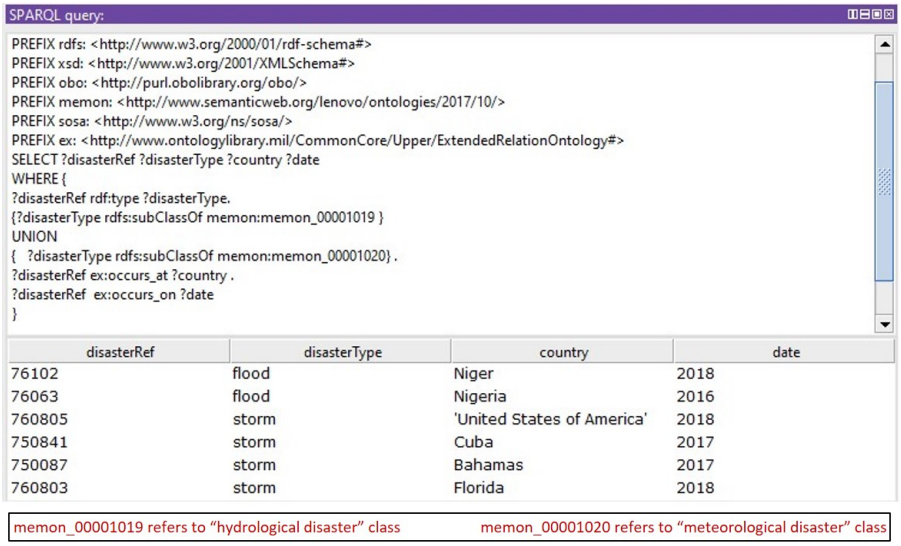


Figure 3.23: SPARQL query script and the results of the natural disasters between 2016 and 2018.

As can be seen in Figure 3.23, two types of natural disasters (floods and storms) are extracted. The result consists of a combination of data from both the EMDAT and the NOAA sources. To obtain information for only one type of natural disasters, we change the "?disasterType" of the SPARQL query to the specific class defined the disaster (`memon_00001019` or `memon_00001020`).

Query 2: What are the amounts of the annual precipitation of the flood events that occurred between 2016 and 2018?

As has been mentioned at the beginning of the chapter, the purpose of the semantic data integration through MEMOn is to provide information from multiple data sources to be able to understand the environmental phenomena. The SPARQL query below asks for the precipitation amounts of the flood disasters that occurred between 2016 and 2018.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX memon: <http://www.semanticweb.org/lenovo/ontologies/2017/10/>
PREFIX ex: <http://www.ontologylibrary.mil/CommonCore/Upper/ExtendedRelationOntology#>
SELECT ?precipitation ?location ?yearMonth ?disasterRef ?disasterType  WHERE {
?precipitation rdf:type memon:memon_00001097 .
?precipitation memon:memon_00001139 ?location .
?precipitation memon:memon_00001145 ?yearMonth .
?disasterRef obo:RO_0000057 ?precipitation.
?disasterType rdfs:subClassOf memon:memon_00001019.
?disasterRef ex:occurs_at ?location.
?disasterRef ex:occurs_on ?yearMonth.
}
```

precipitation	location	yearMonth	disasterRef	disasterType
'61 mm'	Nigeria	201608	76063	flood
'99 mm'	Niger	201808	76102	flood
'92 mm'	Burkinafaso	201808	76080	flood
'80 mm'	Angola	201809	76121	flood

memon_00001097 refers to “precipitation” class
memon_00001019 refers to “hydrological disaster” class
memon_00001139 refers to “observed at” property
memon_00001145 refers to “observed on” property
RO_0000057 refers to “has participant” property

Figure 3.24: SPARQL query and results of the flood disasters and their corresponding precipitation values.

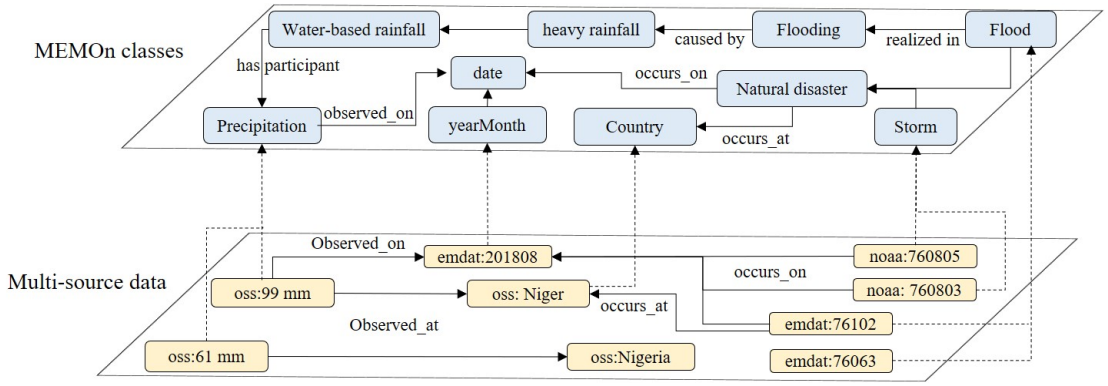


Figure 3.25: Part of the knowledge graph linking multi-source data with MEMOn.

Knowing that the precipitation data is imported from OSS, whereas the data about floods is imported from EMDAT, Figure 3.24 shows the result of this query. Analyzing the query, we can note, firstly, that the query needs information about both average annual precipitation and flood disasters — the result proofs that MEMOn ensures a semantic data integration and linking in a way that allows more effective knowledge retrieval, which are the objectives of MEMOn. Secondly, the result of this query illustrates how we can navigate in MEMOn ontological model to extract the most implicit knowledge. The query's result is obtained by covered relationships, shown in Figure 3.25, including “`realized_in`” relation between “flood” and “flooding” classes, “caused by” link between “flooding” and “heavy rainfall” classes, and other connections such as “`observed_on`” and “`occurs_at`”. Consequently, data can be viewed as a knowledge graph representation due to the relationships among the classes which allow the semantic linking of data. Actually, with our approach, we have not only data semantically integrated, but we also have the correlations between those. With this graph, we could transform information into actionable knowledge as well as extract implicit knowledge.

The inferences are performed on the logical structure of the ontology and of their constituent definitions. The goal of this global data view (knowledge graph) is to retrieve correlated information and exploit it to learn from it and prevent similar events in the future. The primary benefit of this knowledge graph is the flexibility of the connections among the data. The experts can easily connect new data to infer new knowledge according to their expertise and browse links to discover how components/disciplines of the environmental monitoring domain relate to each other.

3.4 Conclusion

In this chapter, we presented a description of the development process of MEMOn, the proposed ontology for the environmental monitoring domain. The proposed ontology offers a semantic representation of the environmental monitoring domain that allows the expressivity of the different environmental disciplines. We adopted the methodology AOM since it is an agile methodology that enabled the incremental and iterative development of a MEMOn's modules in a logical and structured way. MEMOn respects all the fixed requirements initially set. First, it reuses an upper-level and mid-level to improve the interoperability of MEMOn with existing ontologies. Second, it reuses existing domain ontologies to facilitate ontology construction and improve its quality and the efficiency of its maintenance. Third, MEMOn adopts the principle of modularization through the specification of different modules that allow the independence between environmental monitoring disciplines. One of the strengths of this work was relying on real environmental data and involving domain experts for both collecting the vocabulary and testing the ontology. MEMOn is available on-line[6]. It was verified and validated through the data integration framework KARMA to ensure that the proposed ontology supports semantic interoperability, data integration, and data linking to provide a global information view. In chapter 5, MEMOn will be exploited by means of a proposed data integration approach that will enable the semantic interoperability between heterogeneous and multi-source environmental data.

[6]https://github.com/MEMOntology/memon

Chapter 4

Knowledge hypergraph-based approach

Sommaire

4.1 Introduction . . . 97
4.2 Motivating example . . . 98
4.3 Architecture of the proposed approach . . . 100
4.4 Hypergraph-based virtual data integration . . . 101
4.4.1 Semantic annotation . . . 102
4.4.2 RML mappings generation . . . 105
4.4.3 Knowledge hypergraph building . . . 107
4.5 Hypergraph-based query processing . . . 111
4.5.1 Query parsing . . . 112
4.5.2 Hypernodes selection . . . 114
4.5.3 SPARQL sub-queries rewriting . . . 115
4.5.4 Data consolidation and query execution . . . 115
4.6 Conclusion . . . 118

4.1 Introduction

The amount of EO data is not only exponentially large and heterogeneous but also contains implicit relationships. Hypergraphs, which are a generalization of graphs, can be used to better represent earth observations and their relations because of their better expressive capabilities. They can provide a unified and uniform framework to handle the complexities and diversity of EO relationships, especially in the spatiotemporal context. In this chapter, we present a knowledge hypergraph-based approach that (1) virtually integrate EO data using the proposed ontology for the environmental monitoring domain described in the previous chapter and (2) provide a query answering process which aims to enhance queries' results in terms of completeness and relationship richness.

4.2 Motivating example

To highlight the important issues and identify the objectives of our approach, we define a running example to which we shall refer all along with this thesis. We consider three different datasets about precipitation from three different data sources; A raster image in ENVI[1] format, which is a flat-binary raster file provided by OSS, JSON data extracted through the AerisWeather API[2] and a CSV dataset provided by NOAA. This multi-source data presents heterogeneous data schema (metadata), depicted in Figures 4.1 - 4.3. Data schema is presented in bold letters.

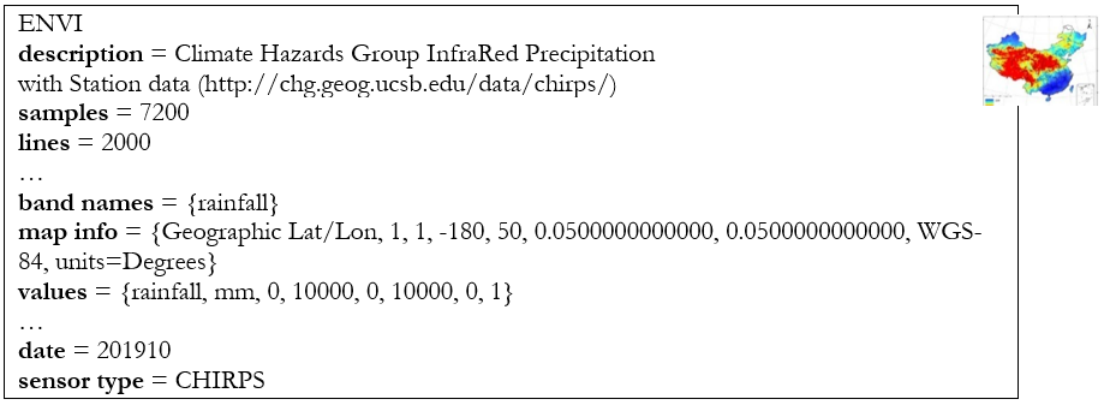


Figure 4.1: Partial view of the metadata from the raster image originating from OSS.



Figure 4.2: Partial view of data in JSON format from AerisWeather API.

We noted the following concerns and issues in the presented datasets. First, various terms are used to describe the same real-world feature that refers to "precipitation" although

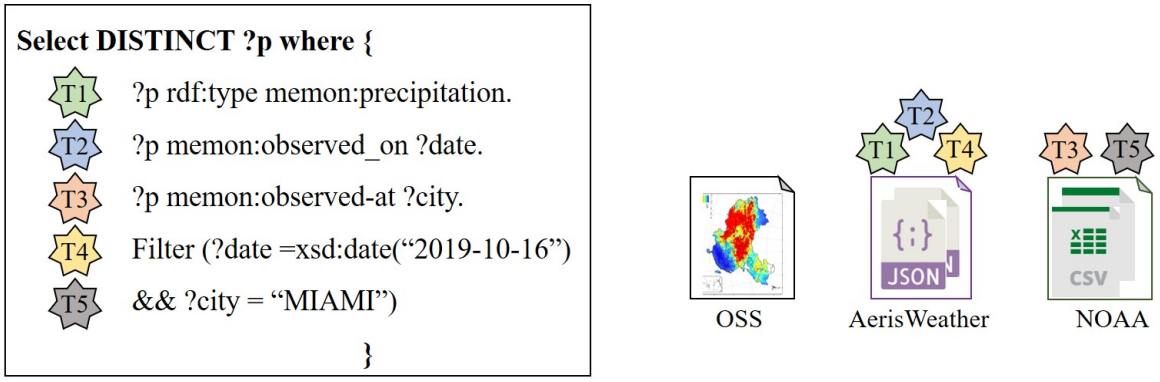
[1]https://www.harrisgeospatial.com/Software-Technology/ENVI
[2]https://www.aerisweather.com/

STATION, CITY, DATE, Hourly Precipitation
COOP:085663, MIAMI FL US,20191016 07:00,0.10
COOP:085663, MIAMI FL US,20191016 11:00,0.10
COOP:085663, MIAMI FL US,20191016 14:00,0.10
COOP:085663, MIAMI FL US,20191016 22:00,0.10
COOP:085663, MIAMI FL US,20191016 23:00,0.10

Figure 4.3: Partial view of data about Hourly Precipitation from the NOAA.

they correspond to the same meaning. For instance, OSS uses the word "rainfall" to refer to "precipitation" in AerisWeather, whereas NOAA uses the term "Hourly Precipitation". The variety of terms complicates the work of experts and software agents, which should be familiar with the terms used in each discipline. Second, the first two datasets list the precipitation values within a location, described by the geographic coordinates (latitude and longitude), whereas, the third dataset contains the precipitation values within a city characterized by the name. Finally, the date of observation is differently represented in the three datasets: "yyyymm" format for the first dataset, "yyyy-mm-jjThh:mm:ss" format for the second dataset, and "yyyymmjj hh:mm." for the third one.

Let us now consider an example of a SPARQL query based on MEMOn ontology, illustrated in Figure 4.4. The query asks for precipitation data observed in "Miami" on 16 October 2019 and comprises five triple patterns T1, T2, T3, T4, and T5.



(a) SPARQL query (b) Data sources

Figure 4.4: Motivating example. (a) SPARQL query over data sources; (b) heterogeneous data sources.

In this example, the AerisWeather API can answer T1, T2, and T4, whereas, NOAA can answer T3 and T5. However, OSS cannot answer any triple pattern since the OSS vocabulary is different from the SPARQL query's graph pattern. Indeed, the query engine relies on source descriptions to select relevant sources for a query. Thus, based on the vocabulary used

in each of the three data sources, the response to this query will not contain a result for several reasons. First, precipitation values are represented differently in the three schemas: by the term "rainfall" in OSS, the term "precipitation" in AerisWeather API and the term "Hourly Precipitation" in NOAA. Second, the spatial representation of the data in OSS and AerisWeather API which is "the geographic coordinates (Latitude and Longitude)," is different from the SPARQL query and the NOAA spatial representation; which is "city." Third, the temporal context of the precipitation observations is represented differently.

Even if the semantic annotation of the data with a domain ontology can resolve the semantic disambiguation between "rainfall" and "precipitation" or between "Hourly Precipitation" and "precipitation", it cannot resolve the difference between the two spatial representations (Lat/Long and city) or the difference between the heterogeneous formats of the date which lies a big issue. Accordingly, the query processing will not guarantee complete results from the different data sources. In other words, traditional query processing cannot ensure finding all results because the data sources contain different data schema.

4.3 Architecture of the proposed approach

We propose a novel knowledge hypergraph-based data integration and querying approach that (1) virtually integrates EO data by maintaining it in their sources and creates a global knowledge hypergraph as a global data catalog and (2) provides a query processing approach that offers an optimal result for SPARQL queries. The architecture of the proposed approach is depicted in Figure 4.5. The layered architecture is composed of four tiers:

1. **The data layer** encompasses different data sources relevant to earth observations and deals with different data formats (e.g., CSV and JSON).

2. **The semantic layer** consists of 3 components, i.e., MEMOn ontology, spatial and temporal RDF stores named `S_RDFStore`, and `T_RDFStore`, respectively. As described in the previous chapter, the modules of MEMOn incorporate different kinds of information entities handling the different contexts of environmental phenomena; spatial and temporal, sensing and observation information that are of importance to the environmental monitoring domain. Also, MEMOn includes various links between its modules to represent relationships between sensing entities, observation entities, and environmental events that they may cause. The two RDF stores (`S_RDFStore` and `T_RDFStore`) are proposed as a solution to the heterogeneity problem of the spatial (resp., temporal) context of EO data. They are aligned to MEMOn ontology and aim to ensure the schematic and semantic interoperability between the different spatial (resp., temporal) representations extracted from the heterogeneous data sources.

3. **The data integration layer** incorporates two main phases: *the hypergraph-based virtual data integration*, and *the hypergraph-based query processing*. The first phase is based on a virtual integration approach of the multi-source and heterogeneous data by building a knowledge hypergraph to ensure semantic interoperability according to the

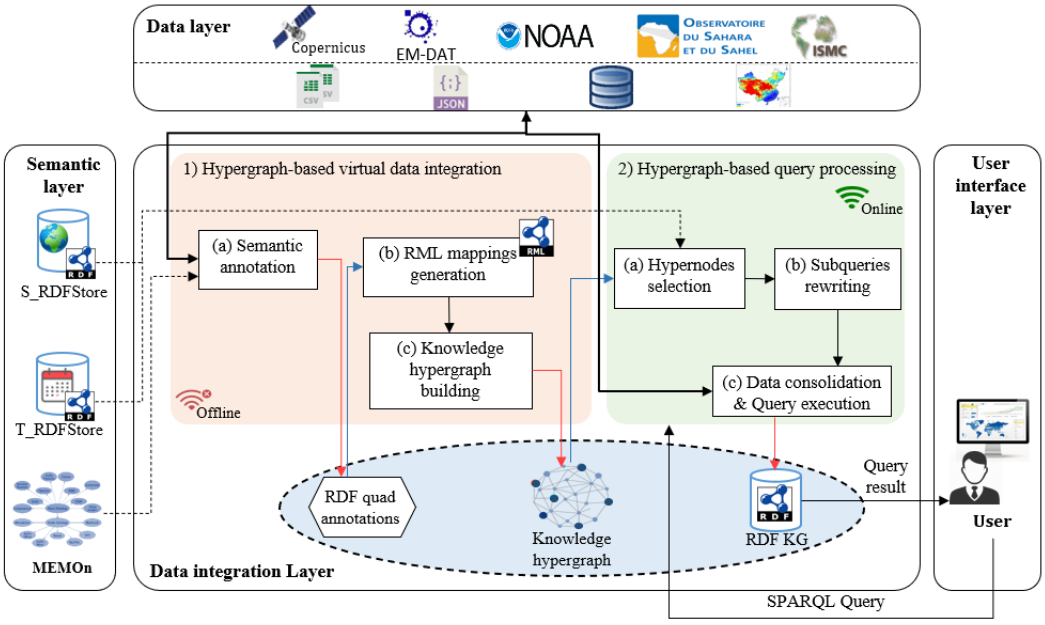


Figure 4.5: Architecture of the proposed knowledge hypergraph-based data integration and querying approach.

OBDI paradigm. It comprises three steps: (a) semantic annotation, (b) RML mappings generation, and (c) knowledge hypergraph building, as shown in Figure 4.5. The second phase consists of extracting and consolidating the appropriate data from various sources as a result of a SPARQL query, on the basis of the knowledge hypergraph built in the first phase. It also comprises three steps: (a) Hypernodes selection, (b) subqueries rewriting, and (c) data consolidation and query execution. In contrast to the first phase, this phase is classified as an online phase since it is only processed after a query input. These layers will be discussed in detail throughout this chapter.

4. **The user interface layer** is a front-end interface allowing the dialog between users and the proposed system. EO engineers, software agents, or even ordinary users, showing adequate knowledge of MEMOn, may have the possibility to query EO data based on a SPARQL queries interface.

4.4 Hypergraph-based virtual data integration

In this section, we introduce the hypergraph-based virtual data integration phase based on the paradigm of OBDI. The process, achieving the vision of OBDI, involves the following three layers:

- **The semantic layer**, representing the ontology. Its goal is to provide a formal and high-level representation of the domain of interest.

- **The data layer**, representing the available data and its metadata.
- **The virtual data integration** based on hypergraphs and representing the mappings between the two previous layers. These mappings are an explicit representation of the relationships between the data sources and the ontology. Their role is to transform a query on the ontology into a query that may be processed by the data source.

The whole process of this phase is illustrated in Figure 4.6. For each dataset, the approach semantically annotates the data using a domain ontology by generating RDF annotations. Then, it generates an RML mapping document that contains mappings between the domain ontology and the metadata, depending on the format of the input (JSON, for example). After that, it builds a mapping view hypernode corresponding to the generated document. Finally, it creates the knowledge hypergraph, composed of mapping view hypernodes and various hyperedges to semantically describe different views of the environmental observations.

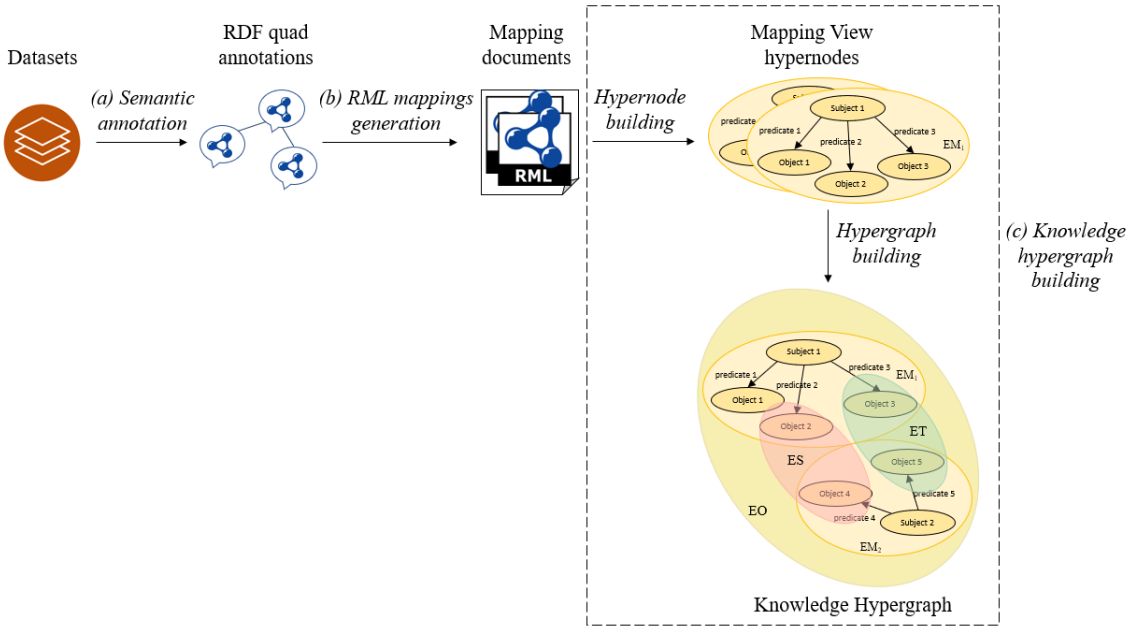


Figure 4.6: The whole process of the hypergraph-based virtual data integration phase.

4.4.1 Semantic annotation

The main idea of the semantic annotation of data (also called semantic matching) is to associate terms used in source data with the classes from the ontology, thus linking together the various resources in a semantically coherent way. In this work, we referred to the Semantic Enhancement (SE) strategy, proposed by Salmen et al., which represents a process for horizontal data integration based on the use of ontologies to integrate and semantically enhance data models (Salmen et al., 2011). This kind of semantic annotation may be characterized

as an "arm's length approach," as it presumes no change of data but rather an association of each piece of data with an entire knowledge base (the ontology). Based on the SE strategy, we define an annotation as follows (Definition 4.4.1).

Definition 4.4.1. An annotation A is a form of metadata attached to a dataset or a specific part of it, like a document or a database field. Each annotation A = < O, C, T, S > has the following components:

(i) O = some ontology class

(ii) T = data term

(iii) C = some relation between O and T (the relation will be rdfs: label)

(iv) S = a reference to the source from which the data term is taken.

Hence, to represent annotations, we choose to use the RDF Quad statements, called also named graphs[3]. One of the main benefits of quads is allowing users to specify various attributes of meta-knowledge that further qualify knowledge, such as the provenance. We update, then, definition 4.4.1 so that annotations with provenance information can be represented in definition 4.4.2:

Definition 4.4.2. Given a set of URI references R, a set of blank nodes B, and a set of literals L, an annotation A is an RDF quad or a four-tuple

$$< \mathrm{O}, \mathrm{C}, \mathrm{T}, \mathrm{S} > \in (\mathrm{R} \cup \mathrm{B}) \times \mathrm{R} \times (\mathrm{R} \cup \mathrm{B} \cup \mathrm{L}) \times (\mathrm{R} \cup \mathrm{B}),$$

Where :

(i) O, C and T are the triple components of RDF, respectively subject, predicate, and object,

(ii) S is the provenance information, and it defines the source from which the data term represented by the object C is taken.

Algorithm 1 presents the steps involved in the semantic annotation. Initially, the system extracts the entities from the metadata (M) (line 8). In the case of structured and semi-structured data (like RDB, CSV, and XML), the proposed approach obtains the corresponding structure information by accessing their schema and extracts the metadata by exploiting the different wrappers according to the data structure. For example, the XML wrapper reads the XSD files that describe the information included in the XML data source. The CSV wrapper reads the CSV file and extracts the column names corresponding to the metadata file.

In the case of raster images, the proposed approach identifies the relevant metadata entities using the two phases, including the extraction of the raster image metadata using the GDAL library and its filtering that aims to limit the extraction of other information that may be non-useful to the integration process.

[3]https://www.w3.org/2009/07/NamedGraph.html#named-graphs

Algorithm 1: Semantic annotation

1 **Input :**
Data
Source : source of the data
C : Set of the ontology's classes
T : Set of the thesaurus' terms
2 **Output :**
Set_A : Set of the RDF Quad Annotations
3 **Variables :**
$M = \{m_1 , m_2 ... m_n\}$: Metadata items
$c \in C$: class from the ontology
$Set_T \in T$: matched terms from the thesaurus
A : An RDF quad annotation
S : Subject of the RDF quad
P : Predicate of the RDF quad
O: Object of the RDF quad
Co : Context of the RDF quad
4 **begin**
5 initialization;
6 $P \longleftarrow$ rdfs:label;
7 $Co \longleftarrow$ source;
8 $M \longleftarrow$ extractMetadata(data);
9 **foreach** *metadata item* $m_i \in M$ **do**
10 **if** *Match* (m_i, C) **then**
11 $c \longleftarrow find(m_i, C)$;
12 **end**
13 **else**
14 $Set_T \longleftarrow find(m_i, T)$;
15 $c \longleftarrow find(Set_T, C)$;
16 **end**
17 $S \longleftarrow c$;
18 $O \longleftarrow m_i$;
19 A = createQuad (S, P, O, Co);
20 $Set_A \longleftarrow Set_A + A$;
21 **end**
22 **end**

Once the metadata entities are extracted, the algorithm exploits the domain ontology as a knowledge base to obtain the semantic entities corresponding to the metadata entities. To do so, each entity of the metadata (m_i) is mapped to one class from the ontology (c) (line 11). If no matching is identified between mi and C, the algorithm exploits the thesaurus, loaded at the beginning of the process, to determine the semantically similar attributes. A set of thesaurus entities that match with the metadata entity are extracted and stored in "set_T" Then, the algorithm matches each thesaurus entity with the ontology classes and extracts the

first corresponding class (line 15). After that, it generates the annotation A comprising the metadata entity m_i, the class c, and the data source Co (line 19).

The whole process is executed to all metadata entities of the input dataset. Accordingly, the result of the first algorithm corresponds to the appropriate set of RDF quad annotations (Set_A), that semantically annotates the data.

4.4.2 RML mappings generation

The proposed approach can automatically produce an RML mapping document that can be used later to generate an RDF graph that corresponds to the input dataset. In this step, the process takes as input one dataset such as a block of a CSV file or an ENVI raster image and produces one RML mapping document as output, using the domain ontology and the annotations produced in the previous step. As outlined in Section 1.3.3, An RML mapping defines a mapping from any data to RDF. Particularly, an RML mapping represents a specification of relations between data schema (metadata entities) and a semantic vocabulary and/or ontology.

We classify the metadata entities into two kinds: simple and complex. Simple entities are defined when information is presented by only one item (eg., when temporal information is described by the term "date"). A complex entity is a set of simple entities (eg., when temporal information is represented by the year, the month, and the day separately). Algorithm 2 presents the pseudo-code of the RML mapping generation process. Summarize, the algorithm is mainly about the following sub-steps:

1. For each metadata, we create a new triples map.
2. For each triples map, we generate a subject map that defines the rule generating unique identifiers for the resources which are mapped. This subject map will be used as the subject of all the RDF triples that can be generated from this triples map. For this purpose, if the input is a raster image, the system identifies the subject of the triples map from the band name of the image. Otherwise, the system generates the subject from the file name.
3. For each triples map, we generate a number of predicate-object maps. The objects correspond to the metadata entities, and the predicates represent the relations between the metadata entities and extracted from the ontology.

   We introduce two other rules to handle the simple and complex metadata entities:

4. Each simple metadata entity is mapped to a predicate-object map and an OWL data- or object-property using an rml:reference.
5. Each complex metadata entity is mapped to another triples map and an OWL object-property using the predicate-object-map property rr:parentTriplesMap. This rule facilitates the generation of more complete mappings.

Algorithm 2: RML mapping generation

Input :
Data
$M = \{m_1 , m_2 \ ... \ m_n\}$: Metadata items
Set_A : Set of RDF Quad Annotations
O : Ontology
Output :
RML_{Map} : RML mapping
Variables :
$M = \{m_1 , m_2 \ ... \ m_n\}$: Metadata items
n = Metadata items number
c: class from the ontology O
r: relation from the ontology O
A_i : The RDF Quad Annotation for m_i
S_{Map} : SubjectMap
PO_{Map} : PredicateObjectMap
begin
  initialization;
  $n \longleftarrow |M|$;
  switch *Data format* **do**
    case *CSV* **do**
      $m_1 \longleftarrow$ read file name(Data);
      break;
    case *ENVI* **do**
      $m_1 \longleftarrow$ read value of the band name(Data);
      break;
    case *Service Web* **do**
      $m_1 \longleftarrow$ read data supplier(Data);
      break;
    otherwise do
      input error;
    end
  end
  TriplesMapName $\longleftarrow$ CreateTriplesMapName(m_1);
  LogicalSource $\longleftarrow$ CreateLogicalSource(Data);
  $S_{Map} \longleftarrow$ CreateSubjectMap(m_1, A_1);
  for $i \leftarrow 2$ **to** n **do**
    // Extract correspondent class to m_i from Set_A
    $c \longleftarrow$ ExtractClass (m_i);
    // Extract the relation between m_i and the correspondent c from the ontology
    $r \longleftarrow$ findRelation(m_i, c, O);
    $PO_{Map} \longleftarrow$ CreatePredicateObjectMap(r, m_i);
    $SetPO_{Map} \longleftarrow SetPO_{Map} + PO_{Map}$;
    $RML_{Map} \longleftarrow print(TriplesMapName,$
    $LogicalSource, S_{Map}, SetPO_{Map})$;
  end
end

4.4.3 Knowledge hypergraph building

The amount of EO data is not only exceedingly large but also contains implicit relationships. Hypergraphs, which are a generalization of graphs, can be used to better represent earth observations and the relations between its various entities because of their better expressive capabilities. Hypergraphs also have the capability of modeling hierarchical and structural forms of data through labeled hyperedges.

RML mappings are themselves RDF graphs and written down in Turtle syntax. In other words, RDF is not only used as of the target data model of the mapping but also as a formalism for representing the RML mapping itself. An RDF graph that represents an RML mapping is called an RML mapping graph consisting of a set of RML triples map (tp), as explained in definition 4.4.3.

Definition 4.4.3. Formally, an RML mapping graph is denoted as `RML_G` = <V,E> Where:

(i) V is a set of vertices representing the subject map and the object maps of a triples map and which correspond to all subjects and objects in RDF data.

(ii) $E \subseteq V \times V$ is a multiset of directed edges that correspond to all triples in RML mapping (predicate maps).

The knowledge hypergraph building step is composed of two sub-steps: hypernodes building and hypergraph building and/or extending[4]. For each RML mapping graph (`RML_G`) and based on the generated RDF quad annotations, we model a semantic view that represents a local linked view of the data source schema, including the classes and relationships derived from the ontology. This semantic view is modeled as a directed graph where nodes are ontology classes, and edges are relationships between these classes. Indeed, to obtain the semantic view over an `RML_G`, the ontology classes corresponding to the subject map and the object maps are represented as nodes, and the ontology relations corresponding to the predicate maps are modeled as edges. We illustrate in Figure 4.7 (a) an example of `RML_G` consisting of two triples maps. The first triples map defines RDF triples of the form (*precipitation,* `observed_at`*, country*) and (*precipitation,* `observed_on`*, yearMonth*). The second triples map defines RDF triples where the subjects come from the first triples map (*yearMonth*) and the objects from the second triples map (*year*) and (*month*). Figure 4.7(b) shows the semantic view, also called mapping view, defined over the `RML_G`. Classes and properties extracted from the `RML_G` are marked with an orange color, whereas those from RDF quad annotations are marked with green color. The mapping views are created over each RML_G. To model these views as a constituent of the hypergraph, we used the concept "hypernodes". A mapping view hypernode defined as follows (Definition 4.4.4).

Definition 4.4.4. Given a hypergraph <V, E>, a **hypernode**[5] is mainly defined as a set of nodes $U \subseteq V$ that act together as a single unit.

[4]When a new data source is added.

[5]Hypernodes may be referred to as undirected hyperedges, compound nodes, or metanodes in the literature.

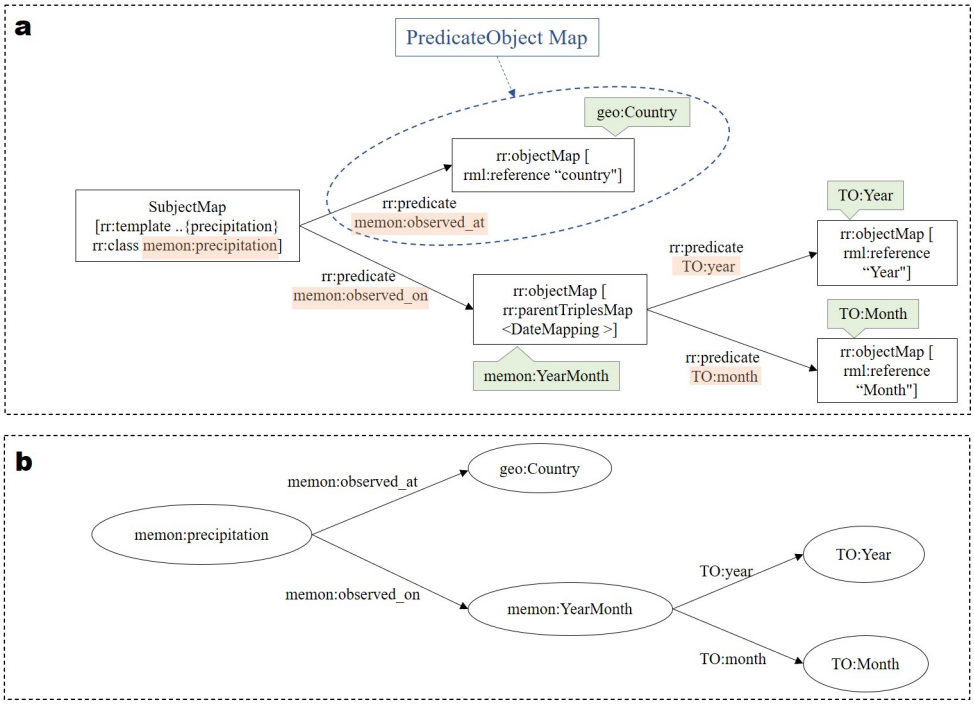


Figure 4.7: (a) An RML mapping graph (`RML_G`). (b) A semantic mapping view over `RML_G`

We defined a **mapping view hypernode** as a directed graph, made up of RDF triples, that we called mapping triples (TM), where nodes represent the classes that correspond to the subject and object maps of an `RML_G` and edges represent the semantic links between these classes, corresponding to the predicate maps.

Also, there will be complex relations between the various entities of observations schema, which could be best represented by edges that can span across several nodes (hyperedges). Particularly, we focused on the relationships between observations in the spatiotemporal context. Thus, we defined three types of hyperedges, named the Spatial-oriented hyperedges, the temporal-oriented hyperedges, and the observation-oriented hyperedges. Based on the definition of knowledge graph (Ehrlinger and Wöß, 2016), we define a knowledge hypergraph in definition 4.4.5.

Definition 4.4.5. A **knowledge hypergraph** is a hypergraph-based knowledge representation that

(i) mainly describes real-world entities and their interrelations, organized in a hypergraph,

(ii) defines complex structures including classes and relations of entities into a hypernode,

(iii) allows for potentially interrelating hypernodes with each other,

(iv) and describes various views and perspectives, using hyperedges.

The proposed knowledge hypergraph can be formally defined as in definition 4.4.6.

Definition 4.4.6. The knowledge Hypergraph is a generalized hypergraph with directed and undirected hyperedges. It can be designated as the tuple:

$$<\mathrm{V},\ \mathrm{A},\ \mathrm{E},\ \mathrm{E}_D,\ \mathrm{E}_M,\ \mathrm{E}_O,\ \lambda_{label},\ \lambda_v >, \text{ where:}$$

- $\mathrm{V} = \mathrm{V}_s \cup \mathrm{V}_o$ is the set of vertices; V_s is the set of all subjects in the mapping views, and V_o the set of all objects;
- A is the set of arcs, i.e., directed edges, an arc is an ordered pair <i,j>, where i,j ∈ V.
- E is the set of hyperedges. $\mathrm{E} = \mathrm{E}_D \cup \mathrm{E}_O$
- $\mathrm{E}_D = \mathrm{E}_S \cup \mathrm{E}_T$ is the set of hyperarcs, i.e., directed hyperedges. Every hyperarc describes a mathematical function, and the direction of the hyperarc shows whether a vertex plays the domain or the range role in a function. E_D is divided into 2 partitions:
  - E_S is composed of the Spatial-oriented hyperedges/hyperarcs. The Spatial-oriented hyperedges can represent collections of spatial representations. The vertices of these hyperedges $\subseteq \mathrm{V}_o$ and refer to the objects representing the spatial context of the observation.
  - E_T is composed of the temporal-oriented hyperedges/hyperarcs. The temporal-oriented hyperedges can represent collections of temporal representations. The vertices of these hyperedges $\subseteq \mathrm{V}_o$ and refer to the objects describing the temporal context of the observation.
- E_M consists of the mapping views represented as hypernodes. Each $\mathrm{e}_m \in \mathrm{E}_M$ is a simple hypernode, i.e., containing only vertices (V) and directed edges (A).
- E_O is composed of the observation-oriented hyperedges. E_O is made up of undirected hyperedges, where each $\mathrm{e}_o \in \mathrm{E}_O$ embodies the hypernodes sharing the same subject.
- $\lambda_{label} : \mathrm{E} \mapsto \mathrm{S}$ is a hyperedge-labeling function. Given a hyperedge e ∈ E, its hyperedge label is deduced according to the pattern of the hyperedge. S is the set of labels.
- $\lambda_v : V^2 \mapsto \mathrm{R}$ is a vertex-transformation-rule function. Given two vertices (v_1, v_2), the transformation-rule is a function that calculates the transformation of an instance of v_1 into an instance of v_2.

From each data source, or rather, each environmental observation (environmental property or phenomenon occurrence), we receive a partial view of information, including a partial view of spatial and temporal representation. To obtain a complete global view of information for an observation, i.e., the entire spatial and temporal representations corresponding to an environmental observation, hypernodes are aggregated. Accordingly, an observation-oriented hyperedge Eo also represents an observation-oriented sub-hypergraph, described by definition 4.4.7.

Definition 4.4.7. The observation-oriented Sub-hypergraph $\mathrm{H}_{Observation}$ consists of:

1. A finite set of vertices U $\subseteq$ V.
2. A finite set of hypernodes, $E_M = e_{m_1}, .. , e_{m_n}$; where $V_{s_i} \in e_{m_i}$ refer to the same class of the ontology, i= 1..n.
3. A finite set of temporal-oriented hyperedges F_T with $F_T \subseteq E_T$ and $F_T \cap E_M$ is not empty.
4. A finite set of spatial-oriented hyperedges F_S with $F_S \subseteq E_S$ and $F_S \cap E_M$ is not empty.
5. An undirected observation-oriented hyperedge $e_o \in E_O$

As an illustration of the observation-oriented sub-hypergraph, an example is illustrated in Figure 4.8 that makes sense of the representation for the mapping view hypernodes into the hypergraph. The essential characteristic is that hypernodes (for example, e_3 in the figure) are themselves vertices in the hypergraph. There are semantic relations among the vertices of the hypergraph. These relations can be described by temporal-oriented hyperarcs (e_2) or spatial-oriented hyperarcs (e_1). The direction of the hyperarc shows whether an object plays the input or output role in a function λ_v. An observation-oriented hyperedge (e4) can be defined as a collection of the mapping view hypernodes whose subjects share the same ontological class.

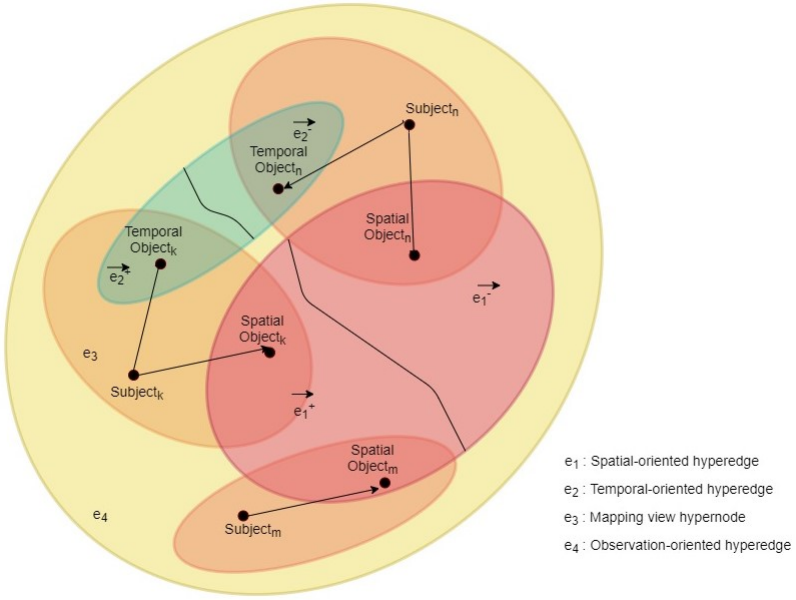


Figure 4.8: Observation-oriented sub-hypergraph describing the relations among observations.

To recapitulate, Figure 4.9 shows an example to explain the whole process of the hypergraph-based virtual integration, starting with the semantic annotation of two different formats of EO data (a JSON and CSV files) until the knowledge hypergraph building. For each input data, the system generates the appropriate RDF quad annotations, constructs the corresponding `RML_G` according to the data schema and the domain ontology used to annotate the data. Then, mapping view hypernodes are extracted from the `RML_Gs` and the annotations. These

hypernodes reflect the central element of the knowledge hypergraph. Also, there are different types of relations among the members of mapping view hypernodes, such as observation, spatial and temporal-oriented relationships.

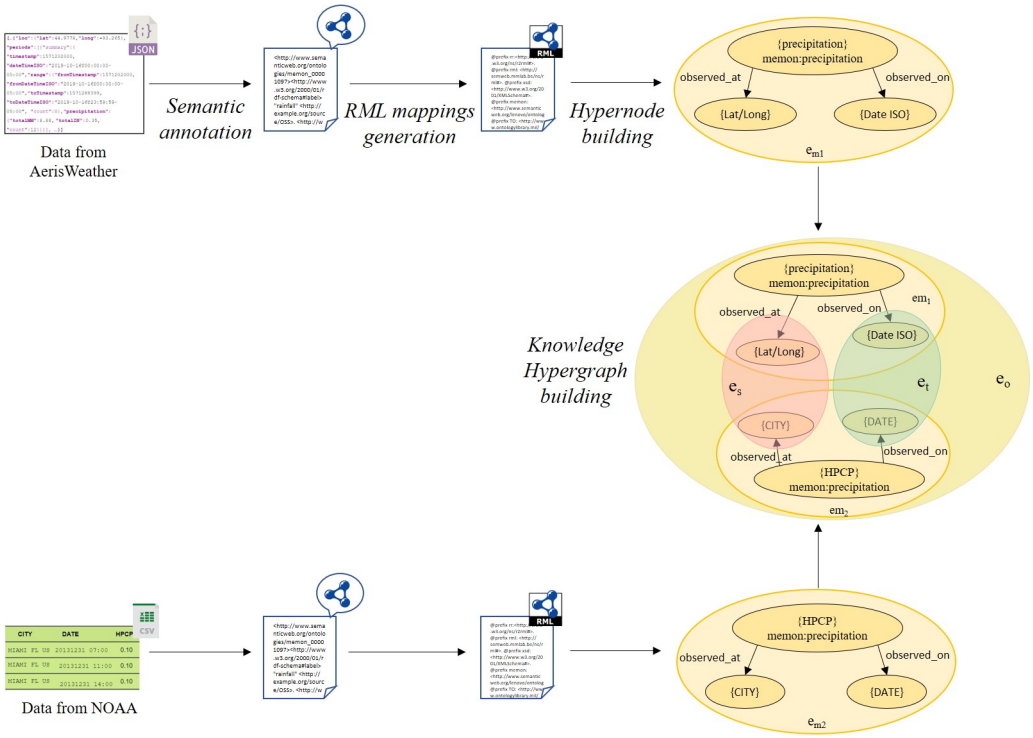


Figure 4.9: The process of the virtual integration from data to knowledge hypergraph building.

4.5 Hypergraph-based query processing

The hypergraph-based query processing phase allows the extraction and the consolidation of data into RDF format through answering SPARQL queries. The whole process takes as input a SPARQL query. Based on the knowledge hypergraph, resulting from the first phase, it generates RDF triples as a result of the input query, and store them in an RDF store. To avoid defining a SPARQL translation method for each target data source query language and to have a specific and a global view answer over all data sources, we introduce a four-step approach, according to the distributed query processing (DQP), discussed in Section 1.4:

(a) The first step "**Query parsing**", consists of parsing the input SPARQL query and generating its schema graph pattern (SGP) using a spatial RDF store and a temporal RDF store.

(b) The second step "**Hypernodes selection**", consists of selecting the corresponding mapping view hypernodes according to the input SPARQL query, given the knowledge

hypergraph. Specifically, the approach matches the SGP with the mapping view hypernodes and extracts a set of relevant mapping view hypernodes and the paths of the RML mapping documents (M_{doc} paths). This step corresponds to the source selection step in the DQP.

(c) Given the extracted set of mapping view hypernodes, the third step, "**Sub-queries rewriting**", consists of transforming the input SPARQL query into concrete sub-queries using the spatial and the temporal RDF stores.

(d) The fourth step, "**Data consolidation and query execution**", involves two sub-steps: The RML mappings processing to generate data in RDF format and store them in buffer RDF stores, and the execution of the sub-queries to obtain the RDF knowledge graph (RDF KG) as a result to the input SPARQL query.

The whole process is depicted and detailed in Figure 4.10.

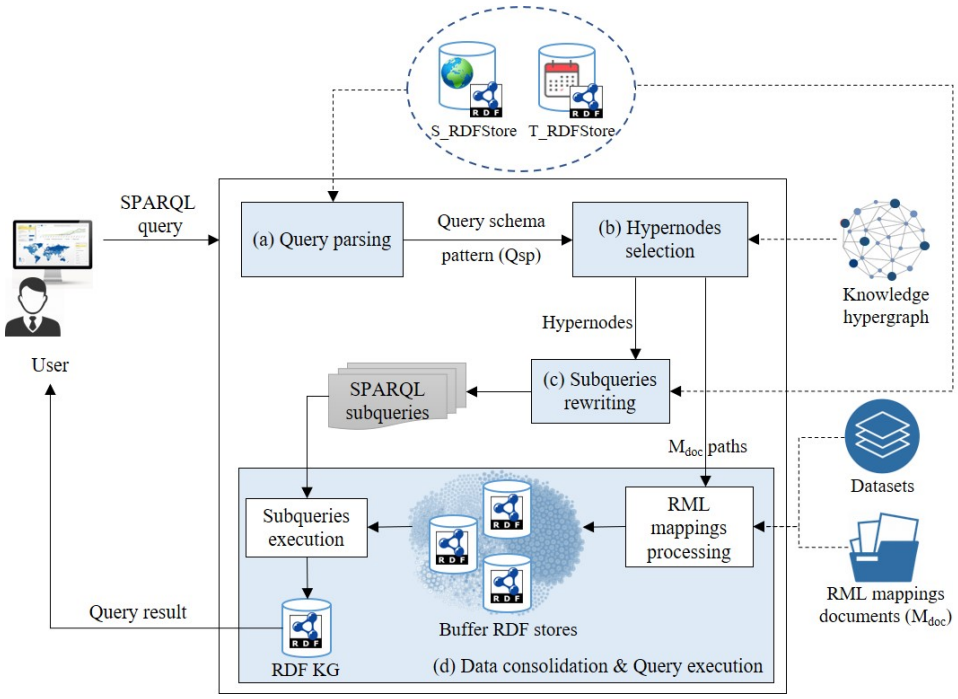


Figure 4.10: The entire hypergraph-based query processing.

4.5.1 Query parsing

The idea of this step is to start with a given SPARQL query (**QR**). The system extracts the SPARQL graph pattern (**GP**) corresponding to the input QR. Then, it generates the schema graph pattern (**SGP**) corresponding to the extracted GP using the domain ontology. In this context, according to Definition 1.2.3, the schema graph pattern is defined as in definition 4.5.1:

Definition 4.5.1. (Schema graph pattern): Given a SPARQL query (QR) and GP its corresponding graph pattern, a SGP is the schema graph pattern where SGP shows only the schema elements corresponding to the keywords in the QR.

$$\text{If } ?x \in \text{var(GP) then SchemaElement}(?x) \in \text{(SGP)}.$$

To illustrate the transformation process of an input QR to SGP, we consider the query in Listing 4.1. It retrieves the precipitation data observed in Miami. The query consists of one GP composed of two triple patterns T1 and T2.

```
SELECT  ?p WHERE {
?p rdf:type memon:precipitation ; # T1
?p :observed_at "Miami". # T2 }
```

Listing 4.1: An example of a SPARQL query.

Figure 4.11 shows the GP of the query above in green and the corresponding SGP in yellow. In order to extract the schema elements corresponding to the instances of GP, we

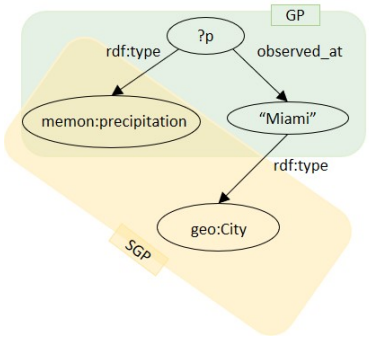


Figure 4.11: The corresponding graph pattern and schema pattern to the query in Listing 4.1

have created a spatial RDF store for the spatial context of data called "`S_RDFStore`" and a temporal RDF store named "`T_RDFStore`" for the temporal context of data. Currently, in our approach, we are only interested in these two contexts.

The aims of the `S_RDFStore` (resp., `T_RDFStore`) are to ensure the semantic interoperability between the different spatial (resp., temporal) representations extracted from the heterogeneous data sources and to enrich the instance represented in the SPARQL query, through an automatic process, with different related relations on spatial (resp., temporal) representations related to the same instance. Thus, the query result can be more enriched.

The `S_RDFStore`(resp., `T_RDFStore`) englobes all the spatial (resp., temporal) representations contained in the different data sources involved in the data integration approach. For example, the term "Miami" is automatically identified as an instance of the "city" ontological class as shown in Figure 4.11, and '20191016' as an instance of the class "date." As a consequence, the process applies the spatial (resp., temporal) context extraction algorithm

to extract other spatial (resp., temporal) representations for the instance "Miami" (resp., '20191016') from the `S_RDFStore` (resp., `T_RDFStore`). Thus, the instance "Miami" is enriched by new relations from the `S_RDFStore` such as:
"Miami" sameAs "Miami_Florida"
"Miami" sameAs "25.7616798° , -80.1917902°".
And the instance "20191016" is enriched by new relations from the `T_RDFStore` such as:
"20191016" year "2019"
"20191016" sameAs "2019/10/16"
"20191016" sameAs "16 October 2019", etc.
Accordingly, the system translates the initial query to sub-queries according to the different temporal and spatial representations.

4.5.2 Hypernodes selection

After generating the SGP, the system matches it with the appropriate mapping view hypernodes from the knowledge hypergraph. Then, it browses in the hypergraph, moving from an hypernode to another along particular hyperedges until it assembles all the targeted mapping view hypernodes. This step is depicted in Definition 4.5.2, in which we elaborate on how to figure out which hypernodes are likely to generate RDF triples matching the SGP.

Definition 4.5.2. Matching of schema SPARQL graph pattern to mapping view hypernodes
Let:

- e_m a mapping view hypernode consisting of a set of mapping triples (TM),
- SGP a schema SPARQL graph pattern composed of a set of triple patterns (TQ).
- the knowledge hypergraph <V, A, E, E_D, E_M, E_O, λ_{label}, λ_v >

We denote by :

- TQ.sub, TQ.pred, and TQ.obj respectively the subject, the predicate, and the object of TQ.
- TM.sub, TM.pred, and TM.obj respectively the subject, the predicate, and the object of TM.

A set of mapping triples TM $\in$ e_m matches SGP if it can produce RDF triples matching TQ of SGP. We denote by setT_{SGP} the set of mapping triples under the knowledge hypergraph that matches SGP. The set setT_{SGP} is defined as follows:

$$\text{set}T_{SGP} = \{\text{TM} \mid \text{TM} \in e_m \wedge (\ \forall\ \text{TQ};\ \text{same(TM.sub, TQ.sub)} \wedge \text{same(TM.pred, TQ.pred)} \wedge \text{match(TM.obj, TQ.obj))}\},$$

where same and match are defined as follows:

- same verifies that the term map and the triple pattern term are the same.
- match (TM.obj, TQ.obj) is valid where TM.obj, TQ.obj $\in$ V $\wedge$ TM.obj $\in$ e $\wedge$ e $\in$ E_D $\wedge$ ($\exists$ v $\in$ V $\cap$ e $\cap$ TM_2) $\wedge$ same(TM.sub, TM_2.sub)

4.5.3 SPARQL sub-queries rewriting

Once the set of hypernodes (E_M) are extracted, the system translates the QR into a union of executable SPARQL sub-queries according to E_M and the logical source description in the RML mapping. Algorithm 3 is a simplified version of this process. It iterates over the set of E_M and replaces each TQ.pred and TQ.obj by TM.pred and TM.Obj, using `T_RDFStore` and `S_RDFStore`. The algorithm implicitly considers the functions between the objects of a specific hyperedge (λ_v). This step is represented in the algorithm by the method "transformate".

Each generated SPARQL sub-query is obtained by matching the SGP with a TM $\in$ setT_{SGP}. The new SPARQL sub-query structure follows the pattern composed of {From, Where}, where:

- "From" let us specify the target triples map's logical source,
- "Where" is a conjunction of conditions on RML data element references, entailed by matching the terms of QR with their corresponding term map in the triples map.

4.5.4 Data consolidation and query execution

The last step of the hypergraph-based query processing phase is the consolidation of appropriate data and the execution of the rewritten sub-queries in order to generate the final result in forms of RDF triples, stored in a triple store.

The process takes as input the extracted RML mapping documents (M_{doc}) and the sub-queries and starts with processing the RML mappings to generate RDF triples. The pseudo-code of the RML mapping processing is provided in Algorithm 4. As a mapping process executor, we opted for RML Mapper[6], which is a Java implementation of an RML mapping processor. The RDF Mapper already supports XML, JSON, and CSV data formats. The process starts by parsing the input mapping and storing it in memory. For each triples map, it opens the data source defined in the logical source and poses the defined iterator query to the data source, using the appropriate library. After receiving the result set, the mapping processor iterates through all features in the results, and for each feature it iterates through all predicate-object maps and processes each one to form the desired RDF triples.

[6]https://github.com/RMLio/rmlmapper-java

Algorithm 3: Translation of Input query to sub-queries

```
1  Input :
     QR : SPARQL query
     Set_{E_M}: Set of hypernodes
2  Output :
     Set_{subQR} : Set of SPARQL sub-queries
3  Variables :
     h ∈ Set_{E_M} : A mapping view hypernode
     sv : A new spatial value
     st : A new temporal value
     subQR ∈ Set_{subQR} : A SPARQL sub-query
4  begin
5      foreach hypernode h ∈ Set_{E_M} do
6          foreach triple pattern tp ∈ h do
7              switch type(tp.predicate) do
8                  case spatial do
9                      if (tp.Object != QR.Object) then
10                         sv= transformate(tp.Object, QR.Object);
11                     end
12                     break;
13                 case temporal do
14                     if (tp.Object != QR.Object) then
15                         st= transformate(tp.Object, QR.Object);
16                     end
17                     break;
18                 otherwise do
19                     break;
20                 end
21             end
22         end
23         subQR = writeQuery(QR, sv, st);
24     end
25     Set_{subQR} = Set_{subQR} + subQR;
26 end
```

Algorithm 4: RML mapping processing

```
1  Input :
     RML_Map : RML mapping
2  Output :
     store : buffered RDF store (RDF triples)
3  Variables :
     map : triples map
     rf = Reference Formulation
     logicalSource: the logical source of the triples map
     RDFTriple : The RDF Triple correspondent to one row
     subjectValue: the subject value of the RDF triple
     predicateValue : The predicate of the RDF triple
     objectValue : The object value of the RDF triple
4  begin
5    foreach triples map ∈ RML_Map do
6      logicalSource = scanLogicalSource ();
7      iterator = ExtractIterator(logicalSource);
8      rf = ExtractReferenceFormulation(logicalSource);
9      logicalReferences = ExtractLogicalReferences(map);
10     switch rf do
11       case CSV do
12         select CSV processor;
13         break;
14       case JSON do
15         select JSON processor;
16         break;
17       otherwise do
18         input error;
19       end
20     end
21     result = ExecuteIterator(iterator);
22     foreach resultRow ∈ result do
23       subjectValue = ExtractSubjectValue(map.SubjectMap, resultRow);
24       foreach predicateObjectMap ∈ map do
25         subjectValue = ExtractPredicateValue(predicateMap, resultRow);
26         objectValue = ExtractObjectValue(objectMap, resultRow);
27         RDFTriple = ConstructRDFTriple(subjectValue, subjectValue,
             objectValue);
28         store = store + RDFTriple;
29       end
30     end
31   end
32 end
```

For each RML mapping document, we obtain a buffered RDF triples store. Then, we execute the sub-queries generated in step 3 over the different RDF stores in order to extract only the RDF triples that match the SPARQL graph pattern (GP) from the first step. To describe the complete approach of the query processing, Algorithm 5 illustrates how the different steps are orchestrated, from the extraction of the GP until the generation of the RDF triples that match it.

Algorithm 5: Knowledge hypergraph querying process

```
1  Input :
     QR : SPARQL query
     K − Hyper: The knowledge hypergraph
2  Output :
     Set_RDFTriples : Set of RDF triples
3  Variables :
     SGP : Schema graph pattern
     HQR : Generated query to interrogate K-Hyper
     Set_spatial: Extracted spatial representations
     Set_temporal: Extracted temporal representations
4  begin
5    SGP = extractSchemaGraphPattern (QR);
6    foreach triple pattern tp ∈ SGP do
7      if (tp.predicate ∈ spatialPredicatesSet ) then
8        Set_spatial = ExtractSpatialRepresentations(SGP.obj, S_RDFStore);
9      end
10     if (tp.predicate ∈ temporalPredicatesSet ) then
11       Set_temporal =ExtractTemporalRepresentations(SGP.obj, T_RDFStore);
12     end
13   end
14   HQR = WriteQuery(SGP, Set_spatial, Set_temporal);
15   Map < Set_EM, paths > = QueryK-Hyper(HQR);
16   Set_subQR = RewriteSubQueries(QR , Set_EM);
17   foreach path ∈ paths do
18     store = RMLMappingProcessing(path);
19     stores = stores + store;
20   end
21   ExecuteSubQueries(stores, Set_subQR);
22 end
```

4.6 Conclusion

In this chapter, we presented a knowledge hypergraph-based approach which is able to virtually integrate heterogeneous data generated from multiple sources and enhance the query answering process in terms of completeness. The proposed approach consists of two phases.

In the offline phase, the system automatically annotates the multi-source data schema and generates the RML mappings that can be used to transform the input data into RDF. Then, it builds the virtual knowledge hypergraph based on the mappings. In the online phase, the system executes a SPARQL query and generates an RDF knowledge graph based on the virtual knowledge hypergraph. It determines a set of relevant hypernodes for the SPARQL triple pattern, rewrites subqueries corresponding to the set of selected hypernode, generates RDF triples for each hypernode, and executes the subqueries to produce an enhanced query result. In the next chapter, we demonstrate the effectiveness of the approach exposed here in terms of accuracy, completeness, and relationship richness.

Chapter 5

Experimentation and evaluation

Sommaire

5.1 Introduction . . . 121
5.2 Onto-KIT tool . . . 121
5.2.1 DISERTO tool . . . 123
5.2.2 HyQ tool . . . 129
5.3 Performance evaluation . . . 132
5.3.1 Experimental setup . . . 132
5.3.2 Experiment 1: DISERTO: semantic annotation evaluation . . . 133
5.3.3 Experiment 2: HyQ evaluation . . . 134
5.3.4 Experiment 3: Comparison with Karma . . . 135
5.4 Discussion . . . 137
5.5 Conclusion . . . 139

5.1 Introduction

In this chapter, we firstly describe the Onto-KIT prototype implementation. Then, we provide a use case study in order to assess the feasibility and promote the effectiveness of the proposed tool. Next, we measure the tool's performance in terms of the accuracy of the returned results. In particular, we measure the precision, the recall, and the F1-measure metrics to evaluate the quality of the schema annotation (matching). Then, we evaluate Onto-KIT in terms of completeness, and we compare the tool performance with Karma framework using various example queries.

5.2 Onto-KIT tool

The proposed approach is implemented through a tool named Onto-KIT (Ontology-based Knowledge hypergraph data Integration and querying Tool). Onto-KIT is composed of two software modules: DISERTO (Data Integration Semantic hypERgraph Tool) that implements the knowledge hypergraph-based virtual data integration phase and HyQ (Hypergraph-based

data Querying) that implements the hypergraph-based query processing phase. Onto-KIT[1] is implemented in Java and integrates several open-source libraries:

- OWL API (Horridge and Bechhofer, 2011): to manipulate the ontology.
- RDF4J[2]: a widely used Java framework for developing Semantic Web applications, tools, and servers. It was used to generate RDF quads, manipulate RDF triple stores, and execute the SPARQL queries.
- GDAL: The OGR Simple Features embedded in this Geospatial Data Abstraction Library (GDAL) were used to read multiple raster formats (ENVI, GeoTIFF, etc.).
- RMLMapper: A Java library that executes RML rules to generate data in forms of RDF triples.
- JavaFX API[3]: An open-source client application built on Java, used to create the tool interfaces.

Figure 5.1 illustrates the main interface of the Onto-KIT tool. We tested Onto-KIT over

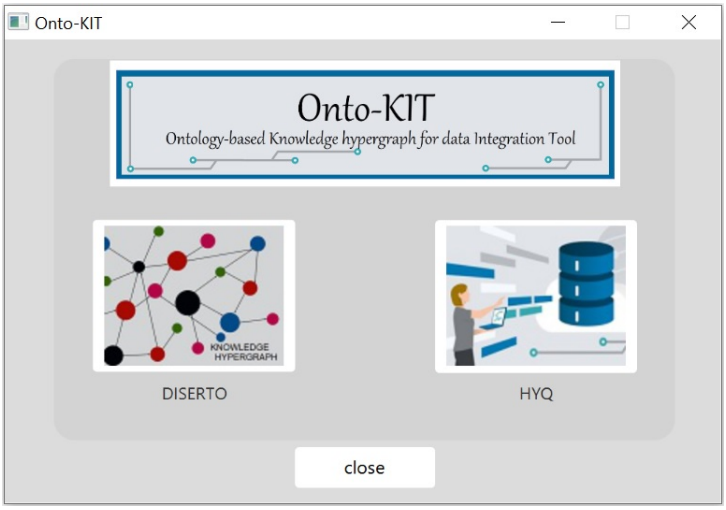


Figure 5.1: Onto-KIT GUI.

real-world datasets to demonstrate the ability of our approach to guarantee semantic interoperability, integrate observed data from multiple sources, and enhance the query answering process. The use case study is based on the motivating example presented in Section 4.2. For data sources, we used dataset about floods in CSV format from EMDAT and heterogeneous datasets about precipitation from three different sources:

- Data in ENVI format (The ENVI image format is a flat-binary raster file) from OSS,

[1]https://github.com/marouamasmoudi/Onto-KIT
[2]https://rdf4j.eclipse.org/
[3]https://openjfx.io/

- Data in JSON format from AerisWeather API,
- Data in CSV format from NOAA.

5.2.1 DISERTO tool

For the virtual data integration phase, DISERTO is implemented to automatically generate the knowledge hypergraph (Masmoudi et al., 2019). To do so, we used the RML model[4] (part of the RML Mapper). DISERTO allows selecting (1) the ontology that will be used as a semantic base for all the steps, (2) the thesaurus if needed, and (3) the input dataset(s). Currently, the tool supports the CSV, JSON, and ENVI formats. The entire process of the hypergraph-based virtual data integration phase is executed when we click on the "Generate" button. Figure 5.2 shows the main interfaces of DISERTO. The outputs consist of an RDF quad annotations document, RML mapping documents, and the knowledge hypergraph stored in an RDF4J triple store.

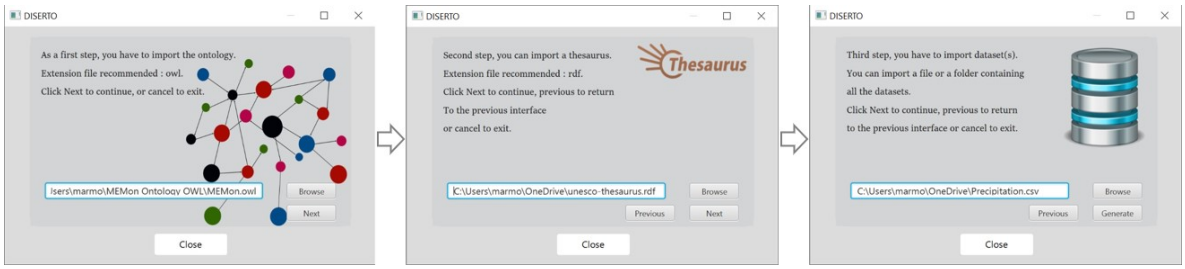


Figure 5.2: DISERTO GUI.

As a first step, we select MEMOn, UNESCO thesaurus, and the datasets, mentioned before. The system loads the ontology and executes the entire process, schematized by the activity diagram, illustrated in Figure 5.3. For each data format, an appropriate wrapper is used to read the data and extract the metadata entities M= $m_{i=1..n}$ (*Step1*). For instance, in the case of CSV file presented in the motivating example (Figure 4.4), the CSV wrapper extracts the columns' names, which correspond to the metadata entities M =(*station, city, date, hourly precipitation*).

For each m_i, the system searches the corresponding class in MEMOn (*Step 3*). MEMOn is checked class by class to find the term m_i. If m_i is found, the corresponding class IRI (c_i) is extracted (*Step 5*). Otherwise, the system refers to the thesaurus (*Step 4*), extracts related terms SetT that it checks with MEMOn classes. Once one term is found in MEMOn, the corresponding c_i is extracted. For instance, for the input ENVI data, the system searches the corresponding c_i to "m_1 = rainfall". Nevertheless, this term is not found in the ontology. Thus, DISERTO searched it in the UNESCO thesaurus, and the related terms of "rainfall" are extracted (rain, precipitation, etc.). Once, the "precipitation" label (which corresponds to the class `memon_00001097`) is found in MEMOn, the RDF quad annotation is generated. While, in

[4]https://github.com/RMLio/RML-Model

JSON data, "m_1 =precipitation" is found in MEMOn as a label to the class `memon_00001097`. Listing 5.1 presents the two annotations generated for these two metadata entities (*Step 6*).

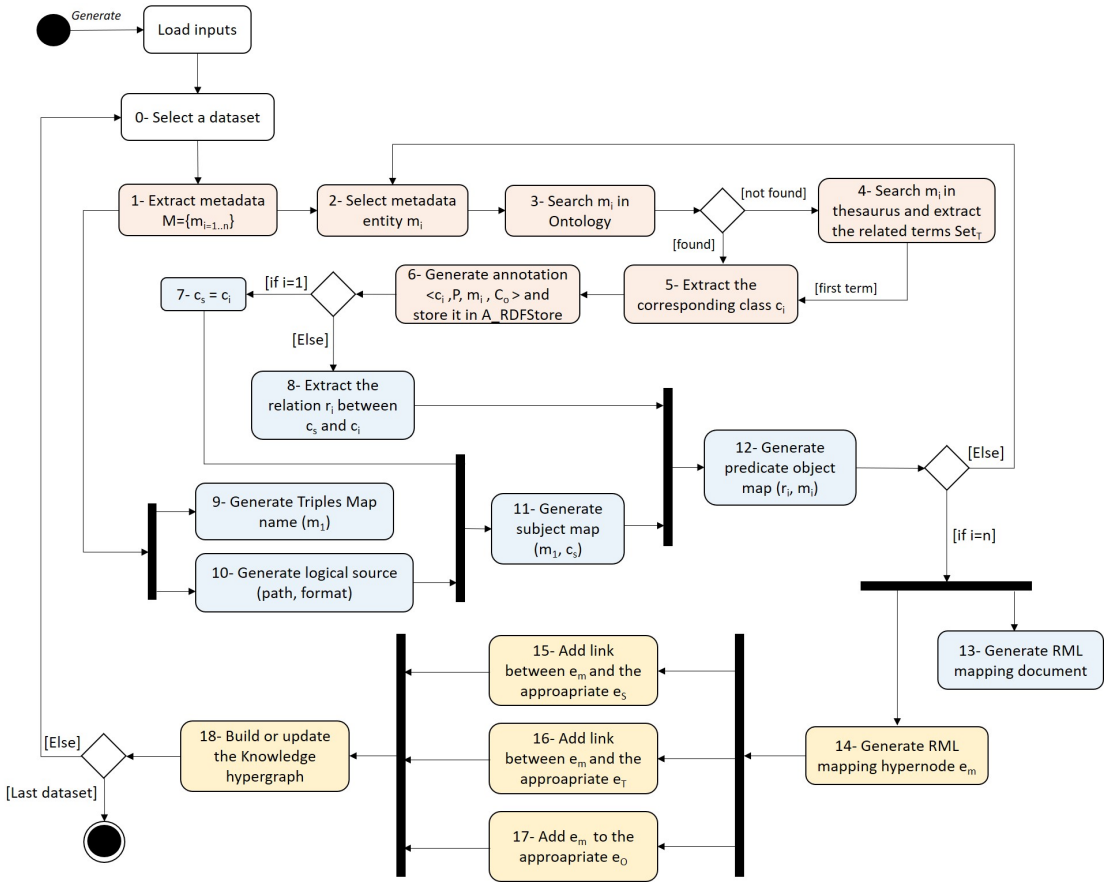


Figure 5.3: Activity diagram of DISERTO.

```
<http://www.semanticweb.org/ontologies/memon_00001097> <http://www.w3.org
    /2000/01/rdf-schema#label> "rainfall" <http://example.org/source/OSS>.
<http://www.semanticweb.org/ontologies/memon_00001097> <http://www.w3.org
    /2000/01/rdf-schema#label> "precipitation" <http://example.org/source/
    Aeris>.
```

Listing 5.1: Two examples of RDF quad annotations.

For each dataset, DISERTO creates an RML mapping document. First, the triples map name is generated from the m_1 value (*Step 9*). Second, based on the data format and its localization, the logical source is generated (*Step 10*). Third, the system generates a subject map to define the rule that generates unique identifiers for the resources which are mapped (*Step 11*). Finally, the predicate object maps are generated (*Step 12*). The number of predicate-object maps depends on the number of relations extracted in *Step 8*. For the JSON dataset, four predicate-object maps are generated as follows:

1. DISERTO generates a predicate-object map that will create a "`memon:observed_on`" link between the subject map and the temporal information in the data.

2. Then, it generates a predicate-object map that will create a "`memon:observed_at`" link between the subject map and the spatial information in the data. The spatial information in the input JSON data is a complex metadata entity (composed of longitude and latitude elements). Accordingly, DISERTO creates a new triples map for the spatial information, named "LatLong." The "`memon:observed_at`" triples will be generated by extracting the subject from the first triples map (<#PrecipitationMapping>), and the objectsfrom the second triples map (<#LatLongMapping>). This can be achieved by adding the *rr:parentTriplesMap* to <#PrecipitationMapping>.
3. For the LatLong triples map, DISERTO generates a predicate-object map that will create a "`geo:has_latitude_value`" link between the subject map and the latitude information in the data.
4. Also, it generates a predicate-object map that will create a "`geo:has_longitude_value`" link between the subject map and the longitude information in the data.

Listing 5.2 illustrates the resulted RML mapping document for the JSON data as automatically generated by DISERTO.

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix rml: <http://semweb.mmlab.be/ns/rml#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix memon: <http://www.semanticweb.org/lenovo/ontologies/2017/10/>.
@prefix TO: <http://www.ontologylibrary.mil/ CommonCore/Mid/TimeOntology#>
<#PrecipitationMapping>
rml:logicalSource [
  rml:source "E:\AerisWeather_data\Precipitation.json";
  rml:referenceFormulation ql:JSON ];
rr:subjectMap [rr:template  "http://example.com/precipitation/{totalMM}";
  rr:class memon:memon_00001097];
rr:predicateObjectMap [
  rr:predicate memon:observed_at;
  rr:objectMap [
    rr:parentTriplesMap <LatLongMapping >] ];
rr:predicateObjectMap [
  rr:predicate memon:observed_on;
  rr:objectMap [ rml:reference "dateTimeISO"] ].
<#LatLongMapping>
rml:logicalSource [
  rml:source " E:\AerisWeather_data\Precipitation.json";
  rml:referenceFormulation ql:JSON ];
rr:subjectMap [
  rr:template "http://example.com/latlong/
  {Latitude},{Longitude}"];
rr:predicateObjectMap [
  rr:predicate geo:has_longitude_value;
  rr:objectMap [ rml:reference "lat" ] ];
rr:predicateObjectMap [
  rr:predicate geo:has_latitude_value;
  rr:objectMap [ rml:reference "long" ] ].
```

Listing 5.2: The resulted RML mapping document for the input Precipitation.json.

After generating RML mapping documents, DIESRTO builds the virtual knowledge hy-

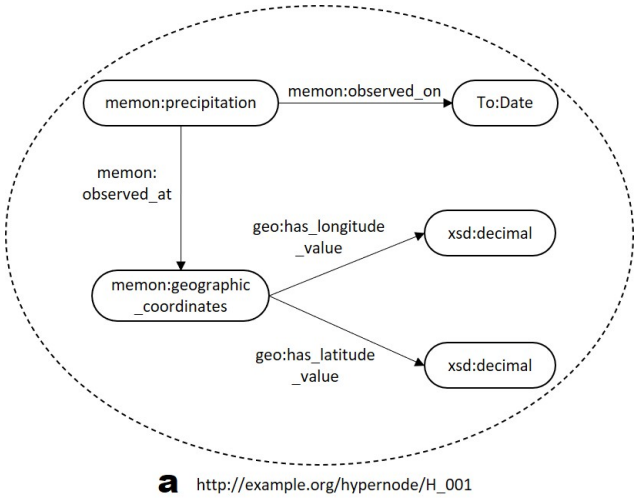


Figure 5.4: The mapping view hypernode "H_001".

pergraph and stores it in an RDF4J store. To do so, for each RML mapping document, the tool produces a mapping view hypernode (e_m) (*Step 14*), then makes it belong to one spatial hyperedge (e_S) (*Step 15*), one temporal hyperedge (e_T) (*Step 16*), and one observation hyperedge (e_O) (*Step 17*). Figure 5.4 shows an illustration of the mapping view hypernode corresponding to the RML mapping document in Listing 5.2. The IRI corrsponding to this hypernode is "http://example/org/hypernode/H_001". Figure 5.5 illustrates the relations between the hypernode "H_001" and its corresponding hyperedges. “b” (resp. “c”) identifies the IRI of the spatial (resp. temporal) hyperedge to which "H_001" belongs. "d" represents the observation hyperedge that includes "a", "b" and "c".

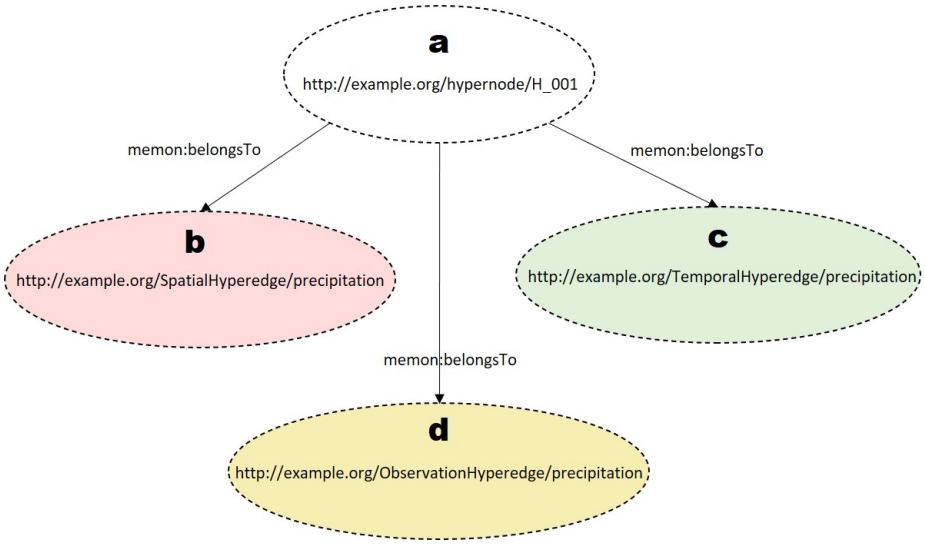


Figure 5.5: The relations between "H_001" and corresponding hyperedges.

To represent the hypernodes and hyperedges, we choose to use RDF quads, where the fourth component of the quad specifies to which hypernode an RML mapping document

corresponds or to which hyperedge, a hypernode belongs. Listing 5.3 represents the generated RDF quads that correspond to Figure 5.5. Figure 5.6 modelizes the four generated RDF quads. The RDF quad "b" identifies the spatial-oriented hyperedge. Specifically, it includes the spatial representation of "H_001" and identifies the IRI of the spatial-oriented hyperedge "*http://example.org/SpatialHyperedge/memon_00001097*", which is automatically generated by the system. Similarly, the RDF quad "c" includes the temporal representation of "H_001" and identifies the IRI of the temporal-oriented hyperedge which corresponds to "*http://example.org/TemporalHyperedge/memon_00001097*". Finally, the RDF quad "d" indicates the path of the RML mapping document corresponding to "H_001" and shows that it belongs to the hyperedge "*http://example.org/ObservationHyperedge/memon_00001097*".

```
(http://www.semanticweb.org/lenovo/ontologies/2017/10/memon_00001097, http://
    www.semanticweb.org/lenovo/ontologies/2017/10/observed_at, http://www.
    semanticweb.org/lenovo/ontologies/2017/10/memon_00001056) [http://example.
    org/hypernode/H_001]

(http://www.semanticweb.org/lenovo/ontologies/2017/10/memon_00001097, http://
    www.semanticweb.org/lenovo/ontologies/2017/10/observed_on, http://www.
    semanticweb.org/lenovo/ontologies/2017/10/memon_00001005) [http://example.
    org/hypernode/H_001]

(http://www.semanticweb.org/lenovo/ontologies/2017/10/memon_00001056, http://
    www.semanticweb.org/lenovo/ontologies/2017/10/belongsTo, http://example.
    org/hypernode/H_001) [http://example.org/SpatialHyperedge/memon_00001097]

(http://www.semanticweb.org/lenovo/ontologies/2017/10/memon_00001005, http://
    www.semanticweb.org/lenovo/ontologies/2017/10/belongsTo, http://example.
    org/hypernode/H_001) [http://example.org/TemporalHyperedge/memon_00001097]

(http://example.org/hypernode/H_001, http://example.org/path, "D:\DISERTO\
    Resources\RML mappings\H_001.ttl") [http://example.org/
    ObservationHyperedge/memon_00001097]
```

Listing 5.3: Examples of RDF quads representing a partial view of the knowledge hypergraph.

Consequently, the knowledge hypergraph generated in the case of our use case study is illustrated in Figure 5.7. It contains 2 observation hyperedges (flood and precipitation), 2 spatial hyperedges, 2 temporal hyperedges, and 4 hypernodes (one for each input dataset). Figure 5.7 also shows the relations among “memon:flood” and a “memon:precipitation” which cover many relationships into the knowledge hypergraph, including “realized_in” relation between “flood” and “flooding” classes, and “caused by” link between “flooding” and “heavy rainfall” classes. Actually, our approach not only integrates data semantically but also determines the correlations between them. Thanks to the knowledge hypergraph, we could transform information into actionable knowledge as well as extract implicit knowledge such as the type of precipitation (heavy, low, etc.). This global data view provided by the generated knowledge hypergraph allows users to retrieve correlated information from data and has the benefit of being exploitable to learn from it and prevent similar events in the future.

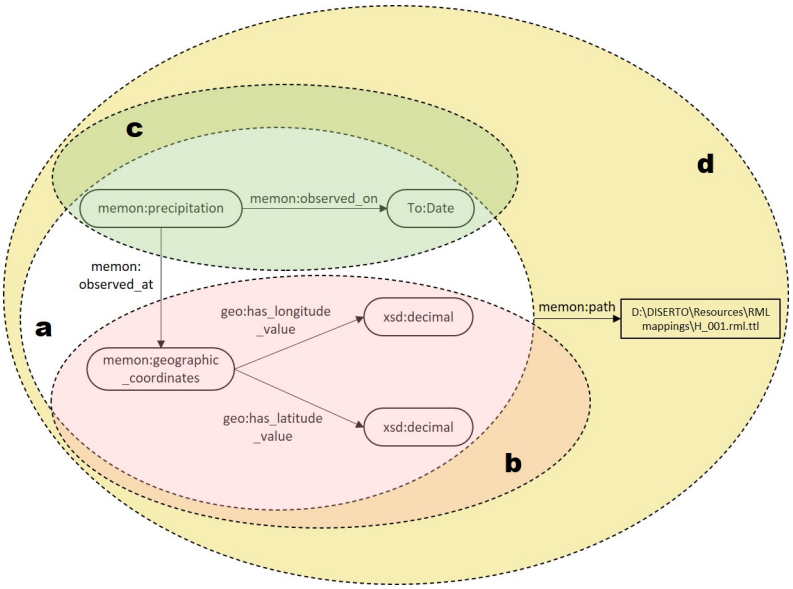


Figure 5.6: A partial view of the precipitation-oriented sub-hypergraph.

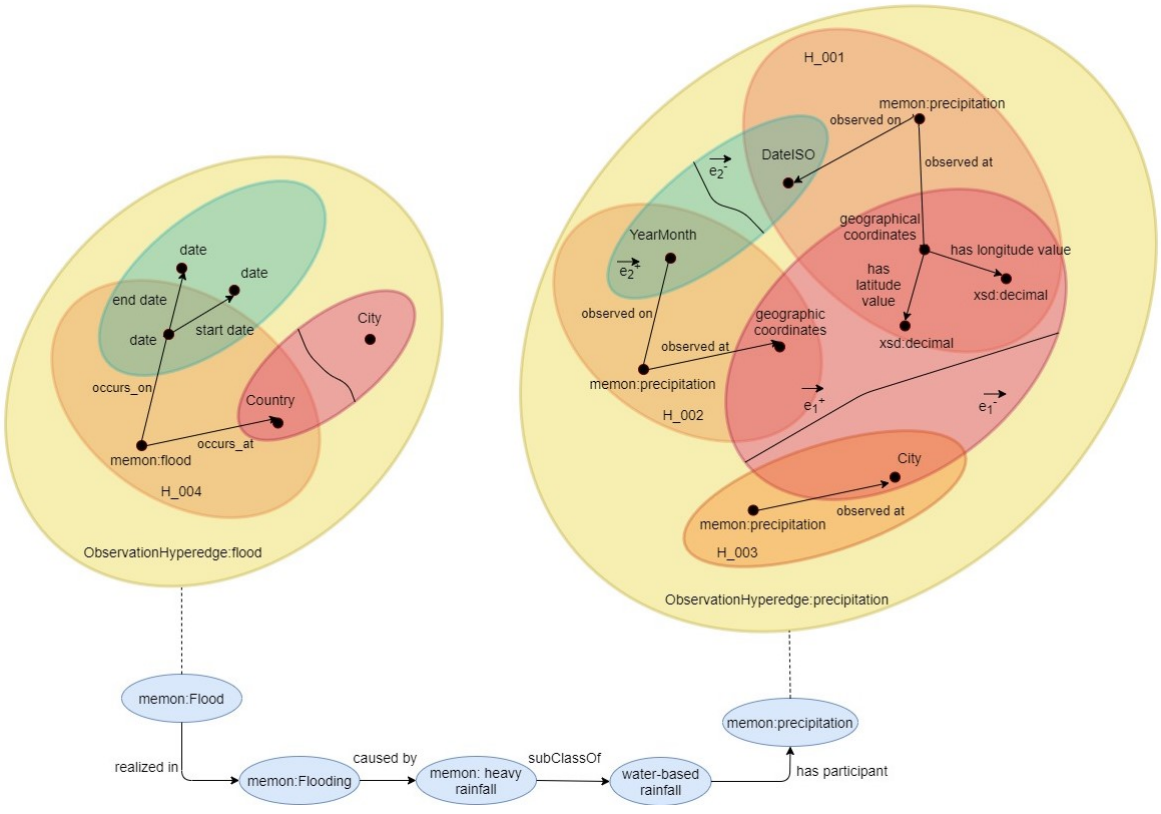


Figure 5.7: A partial view of the knowledge hypergraph.

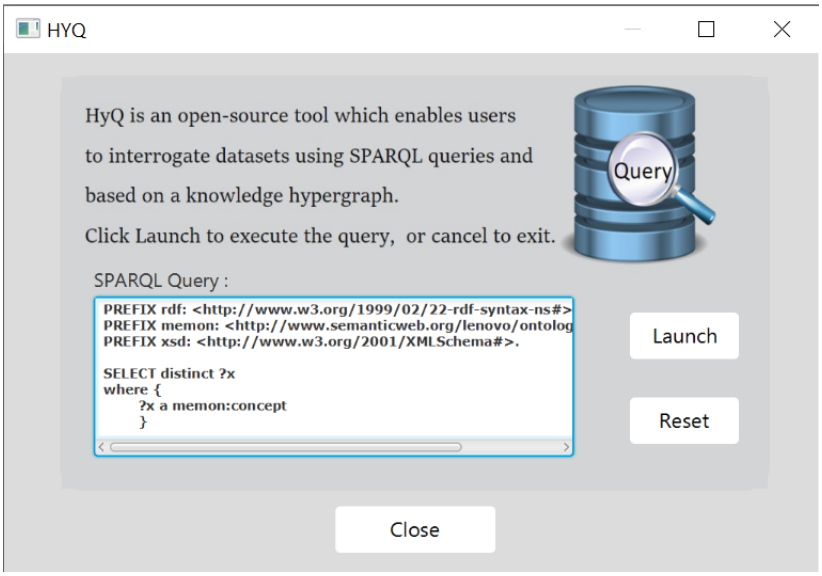


Figure 5.8: HyQ GUI.

5.2.2 HyQ tool

For the hypergraph-based query processing phase, HyQ is implemented to automatically generate the RDF knowledge graph as a result of an input SPARQL query. This process is done based on the knowledge hypergraph generated by DISERTO. HyQ allows (1) entering a SPARQL query through a GUI interface, illustrated in Figure 5.8, (2), extracting the relevant hypernodes and RML mappings through the knowledge hypergraph, (3) rewriting sub-queries, (4) generating RDF triples and (5) executing sub-queries to get an appropriate and optimal result for the input query.

The whole process is illustrated through the sequence diagram (cf. Figure 5.9).

Based on the datasets involved in the use case study, we define the SPARQL query defined in Figure 5.10(a). This query asks for precipitation data observed in "Miami" on “10/2019”.

As a first step, a parsing process is performed. The input query is syntactically analyzed and then translated into a valid syntax for the query engine. To do so, we used the QueryparserUtil of RDF4J. After that, the system extracts the SGP of the input SPARQL query. Figure 5.10(b) shows a graphical representation of the SGP, where arrows represent triples that are oriented from the subject to the object. If the object corresponds to a spatial context, as in this example, the system refers to S_RDFStore to identify “Miami” as a city. To do so, the system automatically generates the query illustrated in Figure 5.10(c) and interrogates S_RDFStore. The result of this query represents the class geo:City. Then, the system matches the SGP with the knowledge hypergraph to extract the corresponding hypernodes. In order to provide an optimal query answering from different data sources, the system relies on S_RDFStore (resp. T_RDFStore) to extract other spatial (resp. temporal) representations that correspond to the representations captured in the input query. Accord-

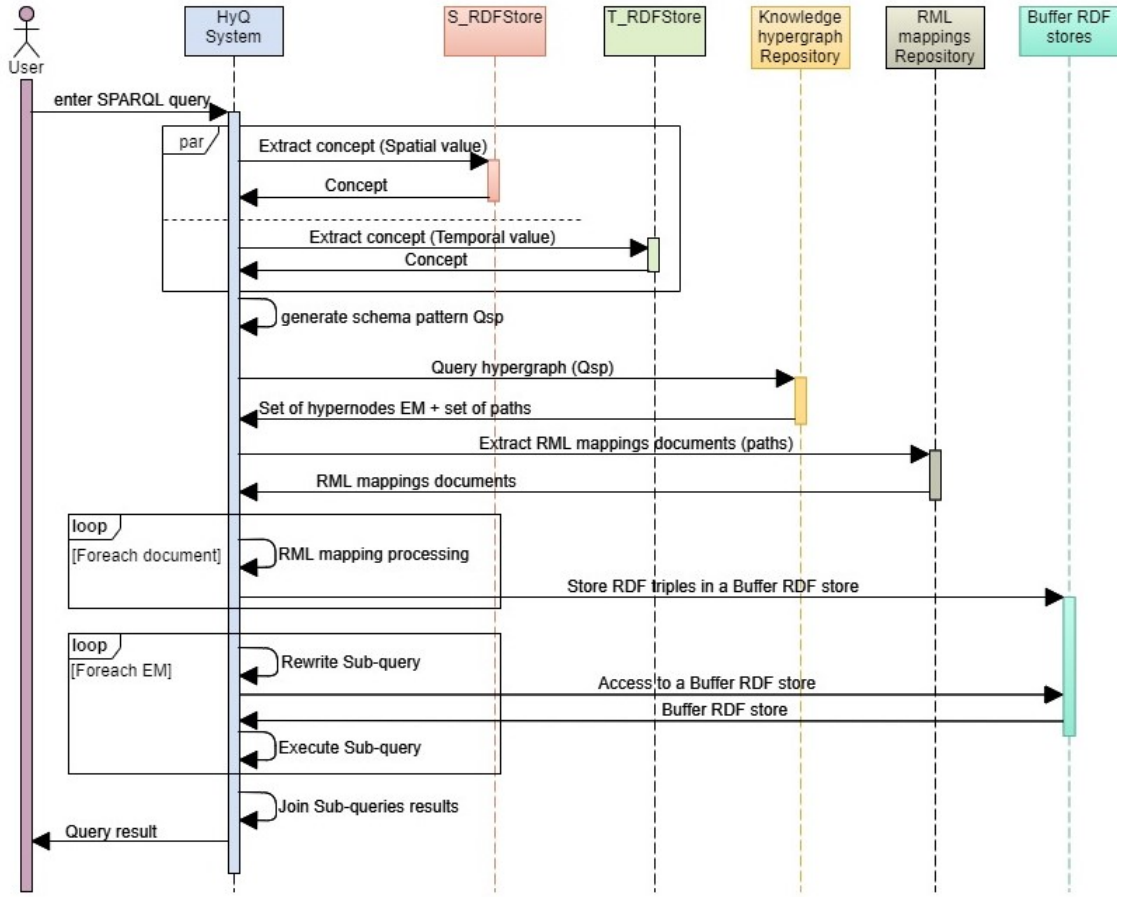


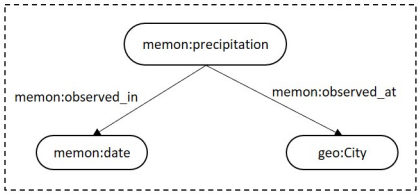
Figure 5.9: Sequence diagram of the query processing.

a

```
PREFIX memon:<http://www.semanticweb.org/lenovo/ontologies/2017/10/>

SELECT ?p ?date WHERE
{ ?p   rdf:type memon:memon_ 00001097; # memon_ 00001097 refers to precipitation
       memon:observed_at "Miami";
       memon:observed_in ?date.
BIND (year(?date) AS ?year)
BIND (month(?date) AS ?month)
FILTER (?year = 2019 && ?month = 10) }
```

b



c

```
SELECT ?type WHERE
{ ?concept rdf:type ?type;
           rdfs:label "Miami"@en.}
```

type
geo:City

Figure 5.10: (a) Example of SPARQL query. (b) The corresponding schema pattern. (c) The query used to interrogate S_RDFStore

a

```
PREFIX memon:<http://www.semanticweb.org/lenovo/ontologies/2017/10/>
PREFIX geo:<http://www.cubrc.org/ontologies/KDD/Mid/GeospatialOntology#>

SELECT ?hypernode ?path WHERE
{ GRAPH <http://example.org/SpatialHyperedge/memon_00001097>
       {geo:City ?predicate ?hypernode}
  GRAPH <http://example.org/ObservationHyperedge/memon_00001097>
       {?hypernode ?predicate2 ?path}
UNION
  GRAPH <http://example.org/SpatialHyperedge/memon_00001097>
       {memon:geographical_coordinates ?predicate ?hypernode}
  GRAPH <http://example.org/ObservationHyperedge/memon_00001097>
       {?hypernode ?predicate2 ?path}
}
```

b

hypernode	path
http://example.org/hypernode/H_001	D:\DISERTO\Resources\RML mappings\H_0001.ttl
http://example.org/hypernode/H_002	D:\DISERTO\Resources\RML mappings\H_0002.ttl
http://example.org/hypernode/H_003	D:\DISERTO\Resources\RML mappings\H_0003.ttl

Figure 5.11: (a) The query used to interrogate the knowledge hypergraph. (b) Results.

ingly, based on the spatial (resp. temporal) -oriented hyperedges, the system interrogates the knowledge hypergraph to identify all relevant hypernodes.The IRIs of the spatial, temporal and observation-oriented hyperedges were automatically identified by the system depending on the variable used in the SELECT part of the SPARQL query pattern (in our case, precipitation). In order to facilitate understanding, Figure 5.11(a) illustrates only the part of the query that concerns the spatial context. Figure 5.11(b) shows the three relevant hypernodes selected by the system.

After that, the system simultaneously generates the RDF triples by processing the RML mappings for each RML mapping document and rewrites the subqueries based on the extracted hypernodes. Each RDF graph is stored in a buffered triplestore.

Given that `H_002` exactly matches the SGP, the system rewrites only two sub-queries for `H_001` and H_003. Listing 5.4 shows the sub-query for both `H_001` and `H_003`.

```
SELECT ?p ?lat ?long ?date WHERE
{ ?p rdf:type memon:precipitation;
memon:observed_at ?latlong;
memon:observed_on ?date.
?latlong geo:has_latitude_value ?lat;
geo:has_longitude_value ?long.
BIND (year(?date) AS ?year) BIND (month(?date) AS ?month)
FILTER (?year = 2019 && ?month = 10 && ?lat >= "30,45" && ?lat <=
   "30,45" && ?long <="-80.192" && ?long >="25.761") }
```

Listing 5.4: The generated sub-query.

The last step of the process of HyQ consists of the execution of the sub-queries on the buffered RDF triple stores, then the union of the results to produce an output RDF knowledge graph as a result of the input SPARQL query. Listing 5.5 shows examples of generated RDF triples :

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
PREFIX memon: <http://www.semanticweb.org/lenovo/ontologies/2017/10/>.
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>.
PREFIX to: <http://www.ontologylibrary.mil/ CommonCore/Mid/TimeOntology#>
PREFIX dbr: <http://dbpedia.org/resource/>

<http://example.com/precipitation/0.10mm> rdf:type memon:memon_00001097.
<http://example.com/precipitation/0.10mm> memon:observed_at dbr:Miami.
<http://example.com/precipitation/0.10mm> memon:observed_on <http://example.com/date
    /20191016>.
<http://example.com/date/20191016> to:year "2019"^^xsd:int.
<http://example.com/date/20191016> to:month "10"^^xsd:int.
```

Listing 5.5: Examples of RDF quads representing a partial view of the knowledge hypergraph.

5.3 Performance evaluation

The objectives of our performance evaluation is twofold: first, we aim at evaluating the accuracy of the semantic annotation step and consequently evaluating the accuracy of the generated RML mappings. Second, we wish to prove that the query processing based on the knowledge hypergraph, proposed in chapter 4, actually enhances the query's results in terms of completeness and relationship richness.

5.3.1 Experimental setup

Environment: We run our experiments on a Windows Intel(R) Core (TM) i5-6300U CPU @ 2.40GHz 2.50 GHz machine, 8,00Go of RAM with Java version 1.8.0.

Metrics:

(i) **Accuracy of the semantic annotation** (semantic schema matching): The quality of schema matching between metadata and ontology classes is commonly measured by precision and recall. While precision measures the number of correctly matched pairs out of all pairs that were matched (cf. equation 5.1), recall measures how many of the actual pairs have been matched (cf. equation 5.2) (Do et al., 2002). However, we can have excellent precision with terrible recall, or alternately, terrible precision with excellent recall. Thus, we also consider the metric F1-measure (cf. equation 5.3) which is the average of precision and recall. We calculate the three metrics using the equations below:

$$Precision = \frac{TP}{FP \cup TP} \tag{5.1}$$

$$Recall = \frac{TP}{FN \cup TP} \tag{5.2}$$

$$F1 - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5.3}$$

With,

- TP: True Positives is the correct set of the automatically derived correspondences,
- FP: False Positives is the set of correspondences falsely proposed by the automatic match operation.
- FN: False Negatives is the set of correspondences needed but not automatically identified.

(ii) **Cardinality:** Number of answers returned by the query.

(iii) **Completeness:** The percentage of the SPARQL query results compared with the expected results.

(iv) **Execution Time:** The elapsed time between the submission of a query to the system and the delivery of the answers.

5.3.2 Experiment 1: DISERTO: semantic annotation evaluation

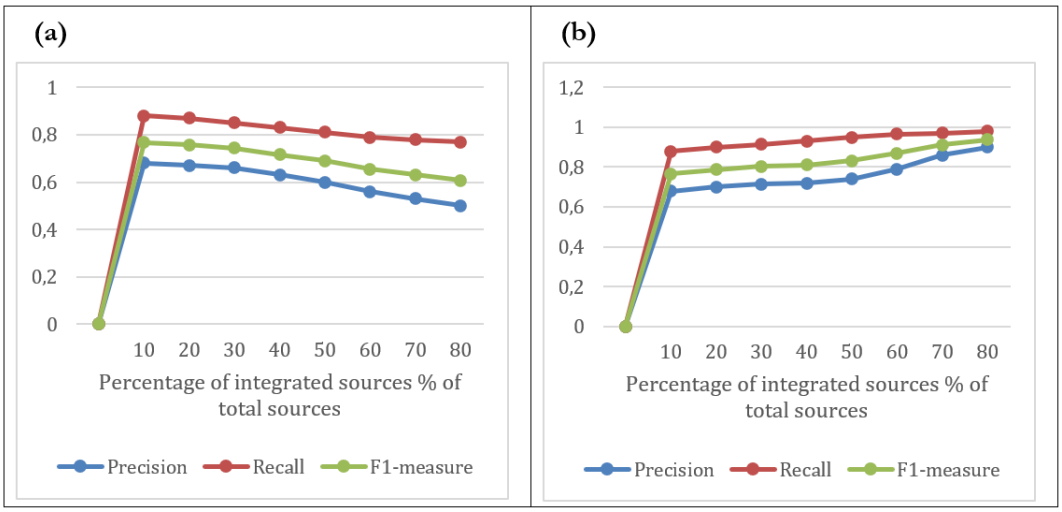


Figure 5.12: Performance variation in terms of precision, recall and F1-measure (a) Only with domain ontology exploitation. (b) with ontology, thesaurus, and annotations exploitation.

Figure 5.12 shows the performance variation in terms of precision, recall, and F1-measure of DISERTO while increasing the number of integrated data sources. Figure 5.12(a) illustrates the experimental measures of the semantic matching step only with the exploitation of the domain ontology. In the second experiment, illustrated in Figure 5.12(b), the system exploits the thesaurus UNESCO and the set of generated RDF quad annotations in addition to the domain ontology while performing the semantic annotation process.

The values of precision and recall that we obtain in the two experiments are greater than 0.5. This is due to the fact that the ontology used to execute the semantic matching contains the key concepts without useless terms (noise) and that it covers well the modeled domain. It can be seen in Figure 5.12(a) that match quality degrades when we integrate other data sources. This decrease can be explained by the fact that the data sources used in this experiment, include more different metadata. In contrast to figure 5.12(b), where match quality increases when we integrate more and more data sources. This illustrates the impact of the exploitation of the thesaurus and the set of annotations in the accuracy of the schema matching and consequently, in the RML mappings. Here, the system attempts to exploit the classes from the domain ontology and the terms from the thesaurus to generate the set of annotations that it stores to be exploited directly next time.

The graph in Figure 5.12(b) shows that after 80% of integrated data sources, the values of the precision and the recall tend toward 1, which means there is less FN and FP returned. Accordingly, the system achieves the best F1-measure of about 0.9. It also shows that exploiting the domain ontology, the thesaurus, and the generated set of annotations yields in new annotations that are significantly more accurate than the annotations generated by only using the knowledge from the domain ontology. In conclusion, thanks to DISERTO, the system accurately annotates the heterogeneous data sources to virtually integrate them into the knowledge hypergraph, which will facilitate the users' query processing to provide them with the desired results.

5.3.3 Experiment 2: HyQ evaluation

In the second experiment, we study the efficiency of query processing when using the knowledge hypergraph. For that, we compute the execution time while scaling up the number of hypernodes. We partition the query processing process into two sub-processes: (1) the steps beginning from the query submission until the hypernodes selection and (2) the steps that include data accessing, RDF triples generation, and query execution. We measured the execution time for each subprocess, then for the entire process. To do so, we run the query three times and retain the fastest execution time. In this experiment, we use the SPARQL query from the motivating example (cf. Figure 4.4). According to the definition of a SPARQL schema graph pattern query (Definition 4.5.1), "var" refers to the variable "precipitation" in our input SPARQL query.

Figure 5.13 reports on the execution time and the percentage of the presence of var (in the example, precipitation) in the knowledge hypergraph, in other words, the presence of hypernodes related to var in the knwoledge graph. The observed results show that the execution time (ET) of the sub-process 1 is almost constant despite the increase in the var presence percentage in the hypergraph. However, the ET of the sub-process 2 rises with the rise of the number of hypernodes related to the SPARQL query's var. We can deduce that the variation (increase/ decrease) of the ET of the entire query processing depends only on the sub-process 2 and particularly on the RDF triple generation step. This means that the time spent in the hypergraph querying and the hypernodes selection does not affect the entire

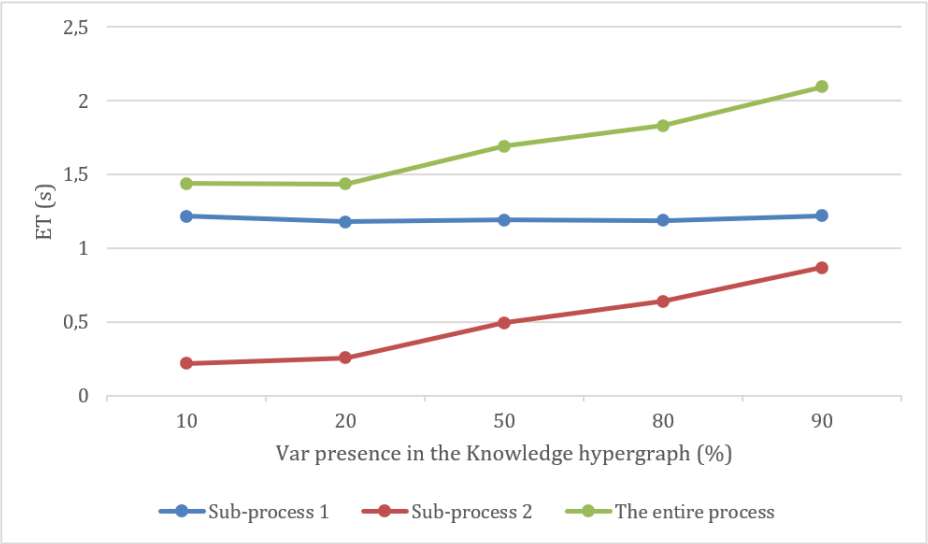


Figure 5.13: Performance variation of HyQ in terms of execution time.

process's ET.

5.3.4 Experiment 3: Comparison with Karma

In this section, we evaluate the efficiency of the proposed data integration and querying approach in terms of performance and results' completeness. Specifically, we define SPARQL queries, execute them, and calculate the cardinality of query responses and the execution time of the query processing. Then, we compare the evaluation results to those of the data integration approach KARMA. As said before, Karma provides a graphical user interface through which we import data, described at the beginning of this chapter (5.2). The output from the interaction with the Karma Web system is a materialized RDF data store used for the querying. We execute four types of SPARQL queries for both systems:

Type 1: Queries that extract variables directly linked to a property without the need for any processing.

Many individual-level factors such as the locations of flood disasters can be extracted with a simple SPARQL query. For example, in MEMOn, the object property 'bfo:occurs_at' was used to link a 'envo:flood' to its 'bfo:site'. Based on this relation, we can use the SPARQL query Q1, as shown in Listing 5.6, to retrieve all floods' locations information, where '?flood' represents the floods and '?site' represents the floods' spatial information.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX memon:<http://www.semanticweb.org/lenovo/ontologies/2017/10/>
PREFIX bfo: <http://purl.obolibrary.org/obo/>
SELECT ?flood ?site WHERE
{
?flood  rdf:type memon:flood;
```

```
                bfo:occurs_at ?site
}
```

Listing 5.6: SPARQL query Q1.

Type 2: Queries that need to process the raw data to produce the desired results.

In EO data sources, different formats are used to describe the temporal information. For example, OSS only considers the month and the year of an event, whereas the raw data in NOAA were recorded as the date of the observation of an event in 'yyyymmdd' format. In our approach, we used the T_RDFStore to link the different formats of date values of an individual of 'bfo:date'. Listing 5.7 illustrates the SPARQL query used in this context.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX memon:<http://www.semanticweb.org/lenovo/ontologies/2017/10/>
PREFIX bfo: <http://purl.obolibrary.org/obo/>
SELECT ?p ?site WHERE
{ ?p    rdf:type memon:precipitation;
        memon:observed_on ?date .
FILTER (?date >= "20191001"^^xsd:date && ?date <= "20191031"^^xsd:date
   }
```

Listing 5.7: SPARQL query Q2.

Type 3: Queries that are used to link a property to spatial context.

In the proposed S_RDFStore, spatial representations are mapped to each other. We use the SPARQL query Q3, defined in Listing 5.8 that retrieves precipitation data in a specific city.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX memon:<http://www.semanticweb.org/lenovo/ontologies/2017/10/>
PREFIX bfo: <http://purl.obolibrary.org/obo/>
SELECT ?p WHERE
{ ?p    rdf:type memon:precipitation;
        memon:observed_at "Niamey" }
```

Listing 5.8: SPARQL query Q3.

Type 4: Queries that generate results based on the knowledge encoded in the ontology.

In section 3.3.4.2, we discussed the usability of the SWRL rules in detail. After integrating the precipitation data from the different sources, a SPARQL query, as shown in Listing 5.9, can be used to retrieve all precipitation data where the precipitation can be described as "very heavy precipitation". To this end, the reasoner can automatically apply the SWRL rules (precisely, the rule R1, presented in Section 3.3.4.1) to the generated buffered RDF stores and deduce the type of precipitation data to ensure that the retrieved precipitation values meet the following condition: precipitation value >16 mm and <50 mm.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

```
PREFIX memon:<http://www.semanticweb.org/lenovo/ontologies/2017/10/>
PREFIX bfo: <http://purl.obolibrary.org/obo/>
SELECT ?p ?city WHERE
{ ?p    rdf:type memon:very_heavy_preciptation;
        memon:observed_at ?city }
```

Listing 5.9: SPARQL query Q4.

Figure 5.14 reports on completeness and cardinality of returned results for the four queries (cf. Listings 5.6 - 5.9) for both Karma and our approach and Figure 5.15 reports on the execution time. In this experiment, we notice that Karma returns responses to queries faster but at the expense of completeness, as can be seen in the chart bar blocks of Q2 and Q3. However, karma fails to answer Q4 which is completely answered by HyQ. HyQ returns complete results for Q1 and Q4 and partially complete results for Q2 and Q3 but slower execution times (>0.8s). In comparison to Karma, HyQ achieves in general higher completeness of results, but at the cost of query execution time (ET). The query processing ET is better at Karma since data integration is not performed at query time but at ETL time. However, it's clearly shown that our proposed approach has much better performance in terms of completeness and relationship richness since Karma approach has not been designed to enhance query response. Contrarily to Karma, HyQ can transform information into actionable knowledge as well as extract implicit knowledge, as illustrated by the case of query Q4.

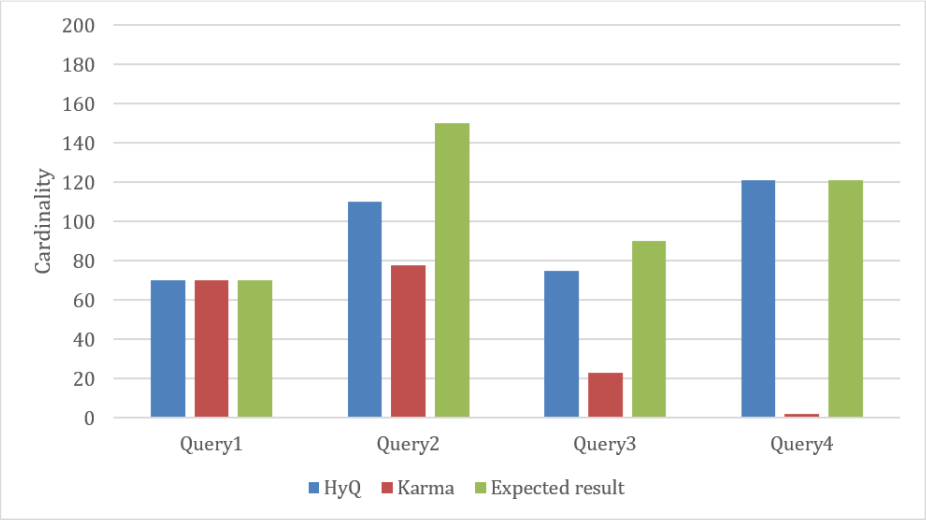


Figure 5.14: Performance variation in terms of completeness and cardinality of returned results.

5.4 Discussion

Onto-KIT aims at effectively addressing semantic data integration and query processing while taking into account the results accuracy, completeness and richness in terms of semantic

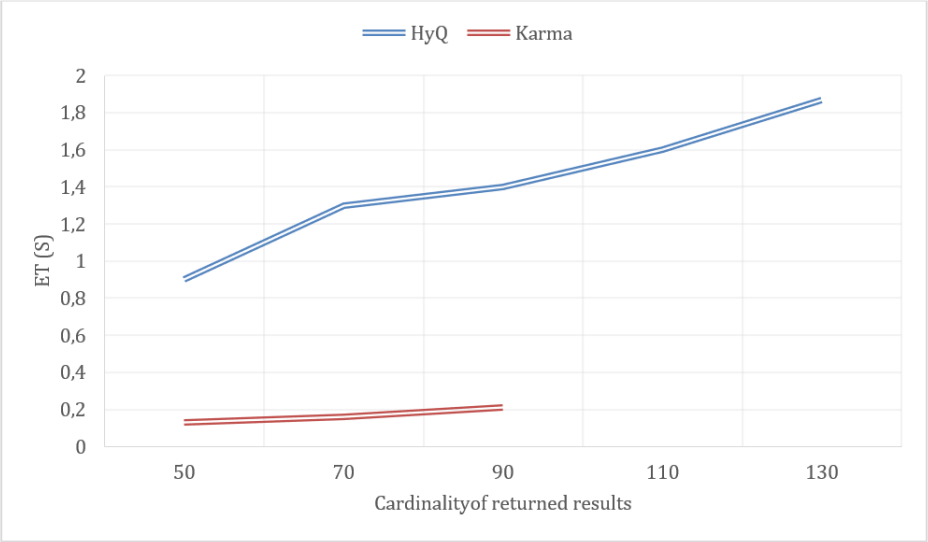


Figure 5.15: Performance variation in terms of execution time and cardinality.

relationships. To achieve this, the proposed tool ensures a virtual integration and linking of data based on a knowledge hypergraph. Specifically, the system employs a domain ontology and a chosen thesaurus in a semantic annotation process to find the corresponding classes that match the terms in metadata and then generates a set of annotations presented as RDF quads. These latter are used along with the domain ontology to create RML mappings. Then, the system builds a knowledge hypergraph (VKHG) that describes data sources in terms of hypernodes with interlinking to other data sources using hyperedges. The generated knowledge hypergraph represents a common information view of distributed data sources and will be used as a data catalog in the query processing. In fact, to produce a complete and rich answer for an input user query, Onto-KIT performs a hypergraph-based query processing through an enhanced source selection step. The source selection mechanism identifies the relevant sources for a specific SPARQL query by referring to the knowledge hypergraph where metadata are semantically defined and linked in a high expressive way. This innovative mechanism aims to cover a broadening spectrum of query response while taking into account results accuracy, completeness, and semantic richness.

Accordingly, the accuracy and completeness metrics play a significant role in measuring the performance of the proposed tool. Thus, we started by measuring the effectiveness of the semantic annotation step, which means measuring whether the tool can fulfill its expectations for schema matching. Next, we evaluated the completeness of the query's answer, including the execution time of the query processing as one of the main comparison criteria.

According to the experimental results, we can deduce that the proposed tool is presenting accurate mappings and significantly enhancing the query answer in terms of completeness by using the hypergraphs as a mathematical structure in the VKHG. Using a hypergraph-based representation to model data schema has an important benefit for expressing relationships. Indeed, results to the query cannot be found when they are part of sources that are unknown

and cannot be discovered during query processing. This is the case when the source descriptions do not match the query and the query processing. no link exists between two sources that may contribute to the final result, but its description in data catalog. As opposed to existing query processing, it might be possible to obtain complete knowledge about different distributed sources. In particular, processing queries against the knowledge hypergraph where sources are linked might yield to more complete results. Although VoID (Vocabulary of Interlinked Datasets) provides description about data sources and links between them, its linkset is limited. Almost predicates such as rdfs:label and owl:sameAs are used. Therefore, VoID lacks details necessary for discovering semantic linking between multi-source data other than equivalence relationships (not limited to label and sameAs). These linking may be expressed in terms of spatial–temporal characteristics, observation characteristics and can be even made richer by incorporating other relations such as the causality knowledge between observations and environmental processes, for example.

However, the cases in which the VKHG contains a mapping that does not cover the entire data source schema have not been addressed. It is believed that the most general case in which the data schema is not entirely covered by the mapping entails new mapping tasks. For example, when data schema contains abbreviations such as HPD (for Hourly Precipitation Data), human intervention is required to map the term with the right semantic class from the ontology.

Regardless of the outpointed limitations, experimental results suggest that the knowledge hypergraph empower the query processing, and allow for the selection of relevant data sources that increase answer completeness. Onto-KIT is available on-line and can be applied to any domain with some adaptations by choosing the appropriate ontology and thesaurus.

5.5 Conclusion

This chapter introduces Onto-KIT, an Ontology-based Knowledge hypergraph data Integration and querying Tool which implements the proposed data integration and querying approach proposed in this thesis. Onto-KIT includes a virtual data integration mechanism that ensures semantic interoperability and semantic integration among heterogeneous data sources using the proposed domain ontology and hypergraphs. This mechanism is implemented in DISERTO . In addition, it comprises a query processing mechanism based on the VKHG generated by the first tool, which is implemented in the HyQ. Experimental results demonstrate that Onto-KIT greatly proves its efficiency in semantically matching and mapping data schema with the domain ontology by generating accurate mappings. Also, it promotes the feasibility of consolidating data as an optimal result to a user's SPARQL query. Using Onto-KIT, makes it possible to integrate heterogeneous and multi-source EO data in order to have a semantic global view of data. It generates accurate semantic mappings and delivers an optimal query's result to the user.

Chapter 6

Conclusions and perspectives

Introduction

In this conclusion, we first provide, in section 6.1, a summary of this thesis as well as some concluding remarks regarding the developed ontology and the proposed semantic data integration and querying approach. Then, we discuss some limitations of this work in section 6.2. Finally, future directions to improve the proposed approach are described in section 6.3.

6.1 Summary

This thesis has made novel contributions to semantic data integration applicated to the environmental monitoring domain. At least four main contributions can be identified in our work: (i) defining a new modular ontology for environmental monitoring domain (Chapter 3), (ii) providing a new semantic data integration approach based on the domain ontology and hypergraphs (Chapter 4), (iii) proposing a knowledge hypergraph-based data querying approach (Chapter 4), and providing a prototype implementation of the data integration and querying solution (Chapter 5). These contributions have answered the research questions defined in the beginning of this work.

Our first research question, in this thesis, was how to formalize the knowledge related to the environmental monitoring domain so as to interpret information semantically. In the literature, formalizing the knowledge of the environment domain was tackled by several research works. However, the existing ontologies deal with specific aspects of the environment; they do not consider the multidisciplinary fields related to environment (meteorology, hydrology, geology, etc.). Accordingly, the first contribution of this thesis is the building of the MEMOn ontology. MEMOn is a modular ontology that provides a common vocabulary of the environmental monitoring domain in order to support semantic interoperability and a global information view of EO data. Chapter 3 detailed the design and the implementation of the set of modules composing MEMOn ontology. These modules embed knowledge about natural disasters, environmental processes, environmental observations and materials, sensors, infrastructure, and spatiotemporal context. It is built from an extensive literature review, in collaboration with the experts of the Observatory of Sahara and Sahel.

The chapter also outlined how this ontology was created according to the AOM agile

methodology and considered a set of best practices, including the use of an upper-level ontology and the reuse of existing ontologies. To link the ontological modules of MEMOn, Basic Formal Ontology was used as the upper-level ontology and Common Core Ontologies as a set of mid-level ontologies. Hence, MEMOn is considered as compliant with other ontologies based on BFO and CCO. In addition, MEMOn reuses terminology and semantics from existing ontologies in the environmental monitoring domain. Indeed, some concepts for environmental processes and observations representation come from ENVO ontology, and some measurements, units, and sensor specifications are drawn from the well-Known SOSA/SSN ontology and OGC standard.

A stated aim of MEMOn was to provide reusable domain ontological modules for other applications. Thus, a number of modules provide full semantic richness representation, with the highest cohesion and poor coupling. While developing MEMOn, we considered the clarity of the ontology's structure by choosing a modular conceptualization since the beginning of MEMOn development process to guarantee ontology evolution and maintenance. The modularity of MEMOn ensures its reuse as separate modules. Though MEMOn was built to represent information about environmental monitoring, the ontology itself can be employed in other domains and can be used in other contexts. For example, soil's experts could use the environmental material module to deal with soil's classification. Besides, the modularity allows the community to expand this work, thus, MEMOn can evolve by adding classes and properties. MEMOn reuses reference ontologies (top-level and mid-level). Consequently, other ontologies can be integrated and reused to enrich MEMOn, by including new modules. For example, a source module that can support information about data sources (source name, data formats, data categories) can be added by reusing the PROV-O ontology. Also, monitoring applications may extend the presented ontology with other modules such as an emergency responder actor module containing actors (e.g., police and firefighters) who may be concerned by environmental monitoring, and the emergency response module including emergency response activities.

Moreover, MEMOn modules can be reused to develop specific domain modules of a relatively-fine granularity. For example, to go into further details in the environmental processes, specific modules, such as volcanic activities or hurricane factors, can be developed. This extension is particularly useful for specific environmental disaster studies. Besides, we can detail environmental material, like water, by adding information about composition, properties, and distribution on Earth. The new module will need classes and relations existing in the environmental material module, observation and measurement module, and geospatial module. Furthermore, given the importance of the AOM methodology that guided us to develop MEMOn, we recommend it for modular ontology developers.

Our second research question was how to break down the data silos and deal with interoperability issues. Accordingly, we proposed a knowledge hypergraph-based virtual data integration approach. This approach exploits MEMOn ontology to resolve the semantic interoperability issue among heterogeneous data sources, and uses hypergraphs to model the global view of local data schemas in order to allow highly semantic expressive linked data. The proposed approach ensures the automation of different tasks such as schema annotation,

mapping generation, and virtual data integration using hypergraphs. We have designed a system's architecture, combining several techniques and results from our literature research, including RML mapping language, RDF quads, RDF stores, etc.

In the semantic annotation process, the system employs the domain ontology and a chosen thesaurus to find the concepts matching the terms from the data source schema and then generates a set of annotations presented as RDF quads. Moreover, the proposed solution simplifies the semantic annotation, as it refers to the generated RDF quads to annotate the data for further annotation processes. As shown in in the first experiment (Section 5.3), using only the domain ontology as the background knowledge does not yield accurate semantic matchings. The reason is that the domain ontology sometimes does not define a large space of possible semantic matchings, and without additional knowledge we cannot resolve this deficit. Fortunately, our approach enables to use additional vocabularies and refers to the generated RDF quads for the next schema matching processes.

Hypergraph was used to build the knowledge hypergraph on the basis of RML mappings. As a mathematical structure, it has allowed the definition of complex objects, using hypernodes and underlined information contexts (spatial, temporal, etc.), using directed hyperedges. Moreover, it stimulated the third research question of our thesis that concerns how to improve the query engine performance in terms of result completeness and relationship richness in order to enrich the returned results and make them more precise and more relevant. Accordingly, we proposed a hypergraph-based query processing approach that enhances the query processing in terms of completeness based on the knowledge hypergraph generated in the data integration. Specifically, the source selection task of the query processing was enhanced to identify relevant sources that possibly contribute to the final result. This was achieved by referring to the knowledge hypergraph, handing the role of a global data catalogue. One most important challenge in a data integration system is the freshness of data. Therefore, having up-to-date data catalogue is a must to ensure fresh data. Nevertheless, updating data catalogue is a costly operation in term of traffic between the data sources and the data integration system. Thus, most of the virtual data integration systems use to create their data catalogues during the query execution which is also an expensive operation. Using the proposed knowledge hypergraph, the integration of a new data source consists merely in the creation of a new hypernode.

The implementation of the proposed data integration and querying approach, which is based on conceptual modelling and RDF implementation, also represents a contribution of this thesis. We have implemented a prototype called Onto-KIT to demonstrate the feasibility and applicability of our proposed solution. We have evaluated Onto-KIT and reported the results of an experimental evaluation carried out in the context of a real-world use case. Analysing the Onto-KIT evaluation' results, we conclude that the proposed solution had a positive impact on the accuracy of the semantic annotation process and the completeness of the query's results.

Furthermore, another benefit of the proposed Onto-KIT tool lies in its ability to be applied to any domain by choosing the appropriate ontology and thesaurus. Unlike Onto-KIT, existing tools (such as GeoTriples and TripleGeo) are specific domain-oriented. They work with Geo

ontologies like GeoSPARQLand Linkedeodata ontologies. Other approaches, such as KARMA are more generic. However, they need a lot of human intervention in data modeling while choosing classes and relations from ontologies. We believe that Onto-KIT constitutes a basis to ensure semantic interoperability through virtual data integration and data linking. It provides an ontology-based knowledge hypergraph that specifies the semantics of the data and links multi-source data. Our tool is an open-source system, and several improvements can be made to increase its efficiency.

6.2 Limitations

We should note that this thesis has some limitations in the methodologies and the capabilities of the adopted technologies. These are discussed below.

When investing in a domain ontology development, the return on the investment is worth and beneficial if the ontology is reused by the community and in several contexts and applications. The use of an upper-level ontology when developing MEMOn enabled this. Indeed, the use of BFO has demonstrated the consistent integration not only among MEMOn modules but also among MEMOn and the BFO-compliant domain ontologies such as ENVO. Consequently, other ontologies based on BFO could be easily integrated. However, in the EO domain, several well-known ontologies non-BFO-compliant, like SWEET ontologies, seem to be helpful to expand the knowledge of MEMOn ontology. Consequently, the integration with those ontologies can be very complicated.

The proposed data integration and querying approach also presents some limitations. Indeed, the schema annotation process exploits a thesaurus chosen by the user to extract relevant classes that do not exist in the domain ontology. One limitation of this contribution is that the approach does not consider the case where neither the ontology nor the thesaurus contains a relevant concept that matches the metadata term. In addition, the RML mappings generation process extracts semantic relationships from the domain ontology to generate the mappings. Accordingly, this process heavily depends on the domain ontology at hand, and this can negatively impact the accuracy of the generated RML mappings. This limitation is not apparent in the evaluation section (Section 5.3), since the domain ontology (MEMOn) used in the experiments is rich in terms of relationships.

Furthermore, the queries processed so far by Onto-KIT tool are still simple; SPARQL filters, as well as join constraints, were not tackled in the translation of an input SPARQL query into sub-queries. The optimization of some SPARQL features such as FILTER EXISTS in distributed queries should be investigated in-depth in order to reap the full benefits of the SPARQL expressiveness.

Concerning the evaluation of the performance of Onto-KIT, it can be questioned if precision and recall are the proper measures. One could argue that recall is not important, as long as there are at least some good hits returned. Also, precision could be irrelevant as long as at least some of the top-ranking hits returned are adequate or satisfactory. Furthermore, the

evaluation of the performance shows that Onto-KIT achieves in general high completeness of results, but at the cost of query execution time. In fact, query processing engines mainly have an adaptive approach rather than designing a one-size-fits-all solution. Thus, in some cases, a choice needs to be made between the challenges of query processing (specifically between completeness and execution time). For instance, it is often not possible to support the query result's completeness and, at the same time, quickly retrieving the results from multiple data sources.

6.3 Perspectives

The outcomes of the research work open some interesting research perspectives towards a more flexible data integration and querying approach. These directions represent further directions that overcome the discussed limitations, and other possible directions raising new questions that could be addressed in further work. In that respect, the following perspectives can be considered.

6.3.1 Ongoing directions

As pointed out in the previous section, the strength of our proposed semantic annotation process is limited when there are not enough relevant semantic concepts in the ontology neither the thesaurus. In future work, the semantic annotation process will be expanded to search concepts in other known semantic models, thesauri, ontologies, and maybe to learn from the generated RML mappings to get more precise mappings. Moreover, other techniques for semantic annotation of source metadata could be investigated. For instance, similarity combination methods may be used to compute correspondences and similarities between metadata terms and ontology classes in order to improve matching quality (Sutanta et al., 2016).

One of the motivations of this thesis is to foster the semantic data linking to identify various relationships, thus contributing to the building of an enriched knowledge graph. Accordingly, we believe that the representation of hyperedges can be made richer than it is now by incorporating other relations such as the causality knowledge between observations and environmental processes, for example. By doing so, we believe that using hypergraph-based learning methods and algorithms, discovering interactions between observations, and generating alerts will be easier in the near future.

As a follow-up activity, we intend to focus on the update of the knowledge hypergraph. Since data sources are always changing, a mechanism responsible for updating only the involved hypernode (s) should be considered.

Another interesting outcome of this thesis is the improvement of query processing in terms of execution time. In future works, we plan to use known SPARQL query federation systems and rely on other approaches and methods that could improve the runtime of the query processing in order to ensure the compromise between query answer completeness and execution

time. Particularly, query optimization techniques shall be investigated to find the efficient query execution plan and optimize for total query execution time. Furthermore, Onto-KIT tool should cover more complex SPARQL queries, expressed with constraints including JOIN, UNION, OPTIONAL and FILTER. Yet, it remains clear that the new implementation of Onto-KIT should consider this limitation for the sake of completeness. Hence, we can evaluate and compare our approach with other data integration and query processing approaches, including the federated systems across different criteria.

Currently, our approach supports JSON, CSV, and ENVI formats in a logical source of the RML mapping. This choice is due to the fact that EO data is generally represented in these formats. Thus, as future work, the new version of Onto- KIT will cover additional data formats such as XML and Shape files. Furthermore, we will concentrate on making our tool scale to even bigger datasets by utilizing big data technologies such as Spark, or HBase and doing more experiments with a larger size of datasets.

Looking forward, Onto-KIT should target more users. In its current state, Onto-KIT provides a simple and clear user interface. Nevertheless, it is targeted only to users familiar with SPARQL language. It should also be easily accessible to non-SPARQL users.

6.3.2 Potential directions

In addition to ongoing directions, this thesis raises new questions that could be addressed in further work. Suggestions for such work are as follows.

As discussed above, using an upper-level ontology ensures consistent modeling of the ontology and promotes its integration with other ontologies based on the same upper-level ontology. However, despite using the same upper-level ontology, knowledge conceptualization may be different from an ontologist to another. For instance, the concept “disaster” is defined in MEMOn as a subclass of “bfo:disposition”. In another BFO-compliant ontology, the same term is classified as a subclass of “bfo:process”. Integrating these two ontologies identified inconsistencies due to disjoint classes between the two “disaster” classes (Elmhadhbi et al., 2020). Accordingly, this integration work raises the need to investigate the integration of existing ontologies in order to improve the methodological framework surrounding the integration of ‘same upper-level ontology’-compliant domain ontologies.

Future work should investigate techniques of alignment between MEMOn modules with other ontologies non-BFO-compliant to guarantee its use by the community in different contexts. Such an extension/direction should not be considered without resolving contextualization issues. Indeed, when knowledge in a particular module is reused by another module, the interpretation of the reused knowledge should be constrained by the context in which the knowledge is being reused. Another direction is to explore context-based alignment techniques to facilitate contextualized interpretation of knowledge by ensuring that the interpretation of assertions in each ontology module is constrained by their context (Xu, 2017).

Alongside this enhanced data integration and querying, a new innovative Decision &

Information architecture should be combined with the technical solution to assist decision making. The proposed approach could be integrated into a Data Value Chain to extract valuable insights (Strange and Zucchella, 2017). By intersecting data with analytics, the Data Value Chain will analyze pre-processed RDF stored data in order to find correlations, identify patterns, and create actionable insights. The analysis could be descriptive by providing insight into the past and identify what has happened, or predictive by analyzing scenarios of what might happen and provide a predictive forecast insight.

To further improve the effectiveness of the proposed approach, the issue of data provenance and value (6th V of Big Data) should be settled (Li et al., 2019). Whilst this thesis assumes that the integrated data sources are reliable, mechanisms and approaches to evaluate the reliability of the data should be investigated.

Bibliography

This bibliography contains 150 References.

Aarnio, P., Seilonen, I. and Friman, M. (2014), Semantic repository for case-based reasoning in cbm services, *in* 'Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)', IEEE, pp. 1–8.

Abbes, H. and Gargouri, F. (2018), 'Mongodb-based modular ontology building for big data integration', *Journal on Data Semantics* **7**(1), 1–27.

Abbes, S. B., Scheuermann, A., Meilender, T. and d'Aquin, M. (2012), Characterizing modular ontologies.

Adeyelu, A. and Anyebe, E. (2018), 'Implementing an improved mediator wrapper paradigm for heterogeneous database integration', *NIGERIAN ANNALS OF PURE AND APPLIED SCIENCES* **1**, 224–235.

Antoniou, G. and Van Harmelen, F. (2004), Web ontology language: Owl, *in* 'Handbook on ontologies', Springer, pp. 67–92.

Arp, R., Smith, B. and Spear, A. D. (2015), *Building ontologies with basic formal ontology*, Mit Press.

Athanasiadis, I. N. (2015), Challenges in modelling of environmental semantics, *in* 'International Symposium on Environmental Software Systems', Springer, pp. 19–25.

Bansal, S. K. and Kagemann, S. (2015), 'Integrating big data: A semantic extract-transform-load framework', *Computer* **48**(3), 42–50.

Beard, K. and Neville, M. (2014), 'A place and event based context model for environmental monitoring', *Context-Awareness in Geographic Information Services (CAGIS 2014)* p. 3.

Benhlima, L. and Chiadmi, D. (2006), 'Vers l'interopérabilité des systèmes d'information hétérogènes', *Electronic Journal of Information Technology* (3).

Bereta, K., Stamoulis, G. and Koubarakis, M. (2018), Ontology-based data access and visualization of big vector and raster data, *in* 'IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium', IEEE, pp. 407–410.

Berners-Lee, T. (2002), 'What do http uris identify?', *Design Issues* .

Berners-Lee, T., Hendler, J. and Lassila, O. (2001), 'The semantic web', *Scientific american* **284**(5), 34–43.

Berners-Lee, T., Hendler, J. and Lassila, O. (2002), 'A new form of web content that is meaningful to computers will unleash a revolution of new possibilities', *Scientific American* .

Botts, M. and Robin, A. (2007), 'Opengis sensor model language (sensorml) implementation specification (ogc 07–000)'.

Bou-Ghannam, A. (2013), 'Redpaper'.

Bretto, A. (2013), 'Hypergraph theory', *An introduction. Mathematical Engineering. Cham: Springer* .

Buttigieg, P. L., Pafilis, E., Lewis, S. E., Schildhauer, M. P., Walls, R. L. and Mungall, C. J. (2016), 'The environment ontology in 2016: bridging domains with increased scope, semantic density, and interoperation', *Journal of biomedical semantics* **7**(1), 57.

Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M. and Xiao, G. (2017), 'Ontop: Answering sparql queries over relational databases', *Semantic Web* **8**(3), 471–487.

Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. and Widom, J. (1994), 'The tsimmis project: Integration of heterogenous information sources'.

Cheatham, M. and Pesquita, C. (2017), Semantic data integration, *in* 'Handbook of big data technologies', Springer, pp. 263–305.

Chen, S., Liu, B. and Rundensteiner, E. A. (2004), 'Multiversion-based view maintenance over distributed data sources', *ACM Transactions on Database Systems (TODS)* **29**(4), 675–709.

Chessell, M., Scheepers, F., Nguyen, N., van Kessel, R. and van der Starre, R. (2014), 'Governing and managing big data for analytics and decision makers', *IBM Redguides for Business Leaders* .

Compton, M., Barnaghi, P., Bermudez, L., GarcíA-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A. et al. (2012), 'The ssn ontology of the w3c semantic sensor network incubator group', *Journal of Web Semantics* **17**, 25–32.

Copernicus (2014), 'Copernicus program', URL: `https://www.copernicus.eu/`.

Corcho, O., Gómez-Pérez, A. and Fernández-López, M. (2004), 'Ontological engineering', *With examples from the areas of Knowledge Management, e-Commerce and the Semantic Web (Advanced Information and Knowledge Processing)* .

Curé, O. and Blin, G. (2014), *RDF database systems: triples storage and SPARQL query processing*, Morgan Kaufmann.

Cyganiak, R., Bizer, C., Garbers, J., Maresch, O. and Becker, C. (2012), 'The d2rq mapping language', *D2RQ Mapp. Lang. D2RQ Platf. URL http://d2rq. org/d2rqlanguage* .

Dahleh, D. and Fox, M. S. (2016), 'An environmental ontology for global city indicators (iso 37120)'.

Das, S., Sundara, S. and Cyganiak, R. (2012), 'R2rml: Rdb to rdf mapping language (2012)', *URL http://www. w3. org/TR/2012/REC-r2rml-20120927* .

Dayal, U. and Hwang, H.-Y. (1984), 'View definition and generalization for database integration in a multidatabase system', *IEEE Transactions on Software Engineering* (6), 628–645.

Devaraju, A., Kuhn, W. and Renschler, C. S. (2015), 'A formal model to infer geographic events from sensor observations', *International journal of geographical information science* **29**(1), 1–27.

Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E. and Van de Walle, R. (2014), 'Rml: a generic language for integrated rdf mappings of heterogeneous data'.

Ding, Y., Lonsdale, D., Embley, D. W., Hepp, M. and Xu, L. (2007), Generating ontologies via language components and ontology reuse, *in* 'International Conference on Application of Natural Language to Information Systems', Springer, pp. 131–142.

Do, H.-H., Melnik, S. and Rahm, E. (2002), Comparison of schema matching evaluations, *in* 'Net. ObjectDays: International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World', Springer, pp. 221–237.

Duschka, O. M., Genesereth, M. R. and Levy, A. Y. (2000), 'Recursive query plans for data integration', *The Journal of Logic Programming* **43**(1), 49–73.

Ehrlinger, L. and Wöß, W. (2016), 'Towards a definition of knowledge graphs.', *SEMANTiCS (Posters, Demos, SuCCESS)* **48**.

Elmhadhbi, L., Masmoudi, M., Karray, M. H. and Archimede, B. (2020), 'Upper-level ontology driven integration of domain ontologies: Application to disaster management', *In 10th international conference on Interoperability for Enterprise Systems and Applications* . In press.

EM-DAT, C. (2012), 'The international disaster database', *Center for Research on the Epidemiology of Disasters. Available at: https://www. emdat. be* .

Emmons, D. L., Bitten, R. E. and Freaner, C. W. (2007), Using historical nasa cost and schedule growth to set future program and project reserve guidelines, *in* '2007 IEEE Aerospace Conference', IEEE, pp. 1–16.

Endris, K. M., Galkin, M., Lytra, I., Mami, M. N., Vidal, M.-E. and Auer, S. (2017), Mulder: querying the linked data web by bridging rdf molecule templates, *in* 'International Conference on Database and Expert Systems Applications', Springer, pp. 3–18.

Erling, O. and Mikhailov, I. (2009), Rdf support in the virtuoso dbms, *in* 'Networked Knowledge-Networked Media', Springer, pp. 7–24.

Falquet, G., Métral, C., Teller, J. and Tweed, C. (2011), *Ontologies in urban development projects*, Springer Science & Business Media.

Fernández-López, M., Gómez-Pérez, A. and Juristo, N. (1997), 'Methontology: from ontological art towards ontological engineering'.

Fernández-López, M., Poveda-Villalón, M., Suárez-Figueroa, M. C. and Gómez-Pérez, A. (2019), 'Why are ontologies not reused across the same domain?', *Journal of Web Semantics* **57**, 100492.

Fox, M. S. (2013), 'A foundation ontology for global city indicators', *University of Toronto, Toronto, Global Cities Institute* .

French-news (2018), 'Floods, aude', URL: `https://www.connexionfrance.com/French-news/What-happened-Meteo-France-explains-record-flooding-in-Aude-despite-storms-` Accessed: January 2019.

Friedman, M., Levy, A. Y., Millstein, T. D. et al. (1999), 'Navigational plans for data integration', *AAAI/IAAI* **1999**, 67–73.

Gangemi, A. (2005), Ontology design patterns for semantic web content, *in* 'International semantic web conference', Springer, pp. 262–276.

Gaševic, D., Djuric, D. and Devedžic, V. (2009), *Model driven engineering and ontology development*, Springer Science & Business Media.

Gialampoukidis, I., Moumtzidou, A., Scarpino, M., Palumbo, G., Vrochidis, S., Kompatsiaris, I., Zaffanella, F., Norbiato, D., Ferri, M. and Vingione, G. (2017), 'Earth observation and social multimedia data fusion for natural hazards and water management: the h2020 eopen project paradigm'.

Glimm, B., Horrocks, I., Motik, B., Stoilos, G. and Wang, Z. (2014), 'Hermit: an owl 2 reasoner', *Journal of Automated Reasoning* **53**(3), 245–269.

Gobin, B. A. (2013), 'An agile methodology for developing ontology modules which can be used to build modular ontologies'.

Görlitz, O. and Staab, S. (2011), Splendid: Sparql endpoint federation exploiting void descriptions, *in* 'Proceedings of the Second International Conference on Consuming Linked Data-Volume 782', CEUR-WS. org, pp. 13–24.

Gray, A. J., Galpin, I., Fernandes, A. A., Paton, N. W., Page, K., Sadler, J., Koubarakis, M., Kyzirakos, K., Calbimonte, J.-P., Corcho, O. et al. (2009), 'Semsorgrid4env architecture–phase i', *Deliverable D1. 3v1, SemSorGrid4Env* .

Gruber, T. R. (1995), 'Toward principles for the design of ontologies used for knowledge sharing?', *International journal of human-computer studies* **43**(5-6), 907–928.

Grüninger, M. and Fox, M. S. (1995), The role of competency questions in enterprise engineering, *in* 'Benchmarking—Theory and practice', Springer, pp. 22–31.

Guarino, N. and Welty, C. A. (2004), An overview of ontoclean, *in* 'Handbook on ontologies', Springer, pp. 151–171.

Hacid, M.-S. and Reynaud, C. (2004), 'L'intégration de sources de données', *Revue Information-Interaction-Intelligence* **3**, 4.

Haller, A., Janowicz, K., Cox, S. J., Lefrançois, M., Taylor, K., Le Phuoc, D., Lieberman, J., García-Castro, R., Atkinson, R. and Stadler, C. (2018), 'The sosa/ssn ontology: a joint wec and ogc standard specifying the semantics of sensors observations actuation and sampling', *Semantic Web* **1**, 1–19.

Henson, C. A., Pschorr, J. K., Sheth, A. P. and Thirunarayan, K. (2009), Semsos: Semantic sensor observation service, *in* '2009 International Symposium on Collaborative Technologies and Systems', IEEE, pp. 44–53.

Hodgson, R., Keller, P. J., Hodges, J. and Spivak, J. (2014), 'Qudt-quantities, units, dimensions and data types ontologies', *USA Available http://qudt. org March* .

Hofweber, T. (2014), 'Infinitesimal chances'.

Horridge, M. and Bechhofer, S. (2011), 'The owl api: A java api for owl ontologies', *Semantic web* **2**(1), 11–21.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M. et al. (2004), 'Swrl: A semantic web rule language combining owl and ruleml', *W3C Member submission* **21**(79), 1–31.

Hose, K., Schenkel, R., Theobald, M. and Weikum, G. (2011), Database foundations for scalable rdf processing, *in* 'Reasoning Web International Summer School', Springer, pp. 202–249.

Hossain, J., Sani, N. F. M., Affendey, L. S., Ishak, I. and Kasmiran, K. A. (2014), 'Semantic schema matching approaches: A review.', *Journal of Theoretical & Applied Information Technology* **62**(1).

InforMEA (2010), URL: `https://www.informea.org/fr`.

IRMA (2017), 'Hurricane irma', URL: `https://response.restoration.noaa.gov/about/media/what-does-sahara-desert-have-do-hurricanes.html`. Accessed: January 2018.

Kapoor, B. and Sharma, S. (2010), 'A comparative study ontology building tools for semantic web applications', *International journal of Web & Semantic Technology (IJWesT)* **1**(3), 1–13.

Karakostas, A., Vrochidis, S., Kompatsiaris, Y., Kantsepolsky, B., Moßgraber, J., Dasiopoulou, S., Mandler, B., Karppinen, A., Ferri, M., Vourvachis, I. et al. (2018), beaware: Enhancing decision support and management services in extreme weather climate events., *in* 'ISCRAM'.

Karray, M. H., Chebel-Morello, B. and Zerhouni, N. (2012), 'A formal ontology for industrial maintenance', *Applied ontology* **7**(3), 269–310.

Katib, A., Rao, P. and Slavov, V. (2017), A tool for efficiently processing sparql queries on rdf quads., *in* 'International Semantic Web Conference (Posters, Demos & Industry Tracks)'.

Katsis, Y. and Papakonstantinou, Y. (2009), 'View-based data integration'.

Kermanshahani, S. (2009), Ixia (index-based integration approach) a hybrid approach to data integration, PhD thesis.

Khan, Z. C. and Keet, C. M. (2016), Dependencies between modularity metrics towards improved modules, *in* 'European Knowledge Acquisition Workshop', Springer, pp. 400–415.

Kharlamov, E., Mailis, T., Mehdi, G., Neuenstadt, C., Özçep, Ö., Roshchin, M., Solomakhina, N., Soylu, A., Svingos, C., Brandt, S. et al. (2017), 'Semantic access to streaming and static data at siemens', *Journal of Web Semantics* **44**, 54–74.

Klyne, G., Carroll, J. J. and McBride, B. (2009), 'Resource description framework (rdf): concepts and abstract syntax, 2004', *February. URL: http://www. w3. org/TR/2004/REC-rdf-concepts-20040210* .

Knoblock, C. A., Szekely, P., Ambite, J. L., Goel, A., Gupta, S., Lerman, K., Muslea, M., Taheriyan, M. and Mallick, P. (2012), Semi-automatically mapping structured sources into the semantic web, *in* 'Extended Semantic Web Conference', Springer, pp. 375–390.

Kollarits, S., Wergles, N., Siegel, H., Liehr, C., Kreuzer, S., Torsoni, D., Sulzenbacher, U., Papez, J., Mayer, R., Plank, C. et al. (2009), 'Monitor–an ontological basis for risk management', *Tech. Rep., Monitor* .

Kyzirakos, K., Savva, D., Vlachopoulos, I., Vasileiou, A., Karalis, N., Koubarakis, M. and Manegold, S. (2018), 'Geotriples: Transforming geospatial data into rdf graphs using r2rml and rml mappings', *Journal of Web Semantics* **52**, 16–32.

Lamy, J.-B. (2017), 'Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies', *Artificial intelligence in medicine* **80**, 11–28.

Laney, D. (2001), '3d data management: Controlling data volume, velocity and variety', *META group research note* **6**(70), 1.

Langegger, A., Wöß, W. and Blöchl, M. (2008), A semantic web middleware for virtual data integration on the web, *in* 'European Semantic Web Conference', Springer, pp. 493–507.

Lenzerini, M. (2002), Data integration: A theoretical perspective, *in* 'Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems', pp. 233–246.

Li, W. C., Nirei, M. and Yamana, K. (2019), 'Value of data: there's no such thing as a free lunch in the digital economy', *US Bureau of Economic Analysis Working Paper, Washington, DC* .

Liu, L. and Özsu, M. T. (2009), *Encyclopedia of database systems*, Vol. 6, Springer New York, NY, USA:.

Llaves, A. and Kuhn, W. (2014), 'An event abstraction layer for the integration of geosensor data', *International Journal of Geographical Information Science* **28**(5), 1085–1106.

Madin, J., Bowers, S., Schildhauer, M., Krivov, S., Pennington, D. and Villa, F. (2007), 'An ontology for describing and synthesizing ecological observation data', *Ecological informatics* **2**(3), 279–296.

Maedche, A. and Staab, S. (2001), 'Ontology learning for the semantic web', *IEEE Intelligent systems* **16**(2), 72–79.

Masmoudi, M., Karray, M. H., Lamine, S. B. A. B., Zghal, H. B. and Archimede, B. (2019), Diserto: Semantics-based tool for automatic and virtual data integration, *in* '2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)', IEEE, pp. 1–8.

Masmoudi, M., Taktak, H., Lamine, S. B. A. B., Karray, M. H., Zghal, H. B., Archimede, B., Mrissa, M., Guegan, C. G. et al. (2018), Predicat: a semantic service-oriented platform for data interoperability and linking in earth observation and disaster prediction, *in* '2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA)', IEEE, pp. 194–201.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A. and Schneider, L. (2002), 'The wonderweb library of foundational ontologies'.

McBride, B. (2002), 'Jena: A semantic web toolkit', *IEEE Internet computing* **6**(6), 55–59. URL: `http://jena.sourceforge.net/`.

McBrien, P. and Poulovassilis, A. (2003), Data integration by bi-directional schema transformation rules, *in* 'Proceedings 19th International Conference on Data Engineering (Cat. No. 03CH37405)', IEEE, pp. 227–238.

Molnár, B. (2014), 'Applications of hypergraphs in informatics: a survey and opportunities for research', *Ann. Univ. Sci. Budapest. Sect. Comput* **42**, 261–282.

NOAA (2014), 'Noaa: National oceanic and atmospheric administration', URL: `http://www.noaa.gov/`.

Noy, N. F. (2004), 'Semantic integration: a survey of ontology-based approaches', *ACM Sigmod Record* **33**(4), 65–70.

Noy, N. F., McGuinness, D. L. et al. (2001), 'Ontology development 101: A guide to creating your first ontology'.

Obrst, L., Ceusters, W., Mani, I., Ray, S. and Smith, B. (2007), The evaluation of ontologies, *in* 'Semantic web', Springer, pp. 139–158.

of the Data Lake, D. V. T. E. (2019), URL: `https://www.ibm.com/cloud/blog/data-virtualization-the-evolution-of-the-data-lake`. Accessed: March 2018.

OGCI, O. (2010), 'Geosparql-a geographic query language for rdf data'.

Oguz, D., Ergenc, B., Yin, S., Dikenelli, O. and Hameurlain, A. (2015), 'Federated query processing on linked data: a qualitative survey and open challenges', *The Knowledge Engineering Review* **30**(5), 545–563.

Oliva-Felipe, L., Gómez-Sebastià, I., Verdaguer, M., Sànchez-Marrè, M., Poch, M. and Cortés, U. (2017), 'Reasoning about river basins: Wawo+ revisited', *Environmental modelling & software* **89**, 106–119.

OSS (2010), 'The sahara and sahel observatory', URL: `http://www.ossonline.org/`.

Özsu, M. T. and Valduriez, P. (1999), *Principles of distributed database systems*, Vol. 2, Springer.

Parsia, B. and Sirin, E. (2004), Pellet: An owl dl reasoner, *in* 'Third international semantic web conference-poster', Vol. 18, Publishing, p. 13.

Patroumpas, K., Alexakis, M., Giannopoulos, G. and Athanasiou, S. (2014), Triplegeo: an etl tool for transforming geospatial data into rdf triples., *in* 'Edbt/Icdt Workshops', pp. 275–278.

Pease, A., Schalley, A. and Zaefferer, D. (2006), 'Formal representation of concepts: The suggested upper merged ontology and its use in linguistics', *Ontolinguistics. How Ontological Status Shapes the Linguistic Coding of Concepts* pp. 174–211.

Protégé (2014), '"stanford university"', URL: `http://protege.stanford.edu/`.

Prud'Hommeaux, E., Seaborne, A. et al. (2007), 'Sparql query language for rdf (working draft)', *Bericht, W3C, March* .

Qiu, L., Du, Z., Zhu, Q. and Fan, Y. (2017), 'An integrated flood management system based on linking environmental models and disaster-related data', *Environmental Modelling & Software* **91**, 111–126.

Quilitz, B. and Leser, U. (2008), Querying distributed rdf data sources with sparql, *in* 'European semantic web conference', Springer, pp. 524–538.

Rakhmawati, N. A., Umbrich, J., Karnstedt, M., Hasnain, A. and Hausenblas, M. (2013), 'Querying over federated sparql endpoints—a state of the art survey', *arXiv preprint arXiv:1306.1723* .

Rani, P. S., Suresh, R. and Sethukarasi, R. (2019), 'Multi-level semantic annotation and unified data integration using semantic web ontology in big data processing', *Cluster Computing* **22**(5), 10401–10413.

Raskin, R. G. and Pan, M. J. (2005), 'Knowledge representation in the semantic web for earth and environmental terminology (sweet)', *Computers & geosciences* **31**(9), 1119–1125.

RDF (2014), 'Resource description framework', URL: `https://www.w3.org/RDF/`.

RDF4J, E. (n.d.), URL: `https://rdf4j.org/`. Accessed: June 2019.

Regueiro, M. A., Viqueira, J. R., Stasch, C. and Taboada, J. A. (2017), 'Semantic mediation of observation datasets through sensor observation services', *Future Generation Computer Systems* **67**, 47–56.

Report (2019), 'World economic forum', URL: https://www.weforum.org/agenda/2019/04/data-oildigital-world-asset-tech-giants-buy-it/.

Saleem, M., Khan, Y., Hasnain, A., Ermilov, I. and Ngonga Ngomo, A.-C. (2016), 'A fine-grained evaluation of sparql endpoint federation systems', *Semantic Web* **7**(5), 493–518.

Saleem, M. and Ngomo, A.-C. N. (2014), Hibiscus: Hypergraph-based source selection for sparql endpoint federation, *in* 'European semantic web conference', Springer, pp. 176–191.

Salmen, D., Malyuta, T., Hansen, A., Cronen, S. and Smith, B. (2011), 'Integration of intelligence data through semantic enhancement'.

Saulnier, D. D., Green, H. K., Ismail, R., Chhorvann, C., Mohamed, N. B., Waite, T. D. and Murray, V. (2019), 'Disaster risk reduction', *Disaster Prevention and Management: An International Journal* .

Sauvé, S., Bernard, S. and Sloan, P. (2016), 'Environmental sciences, sustainable development and circular economy: Alternative concepts for trans-disciplinary research', *Environmental Development* **17**, 48–56.

Schoening, J. R., Duff, D. K., Hines, D. A., Riser, K. M., Pham, T., Stolovy, G. H., Houser, J., Rudnicki, R., Ganger, R., James, A. et al. (2015), Ped fusion via enterprise ontology, *in* 'Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR VI', Vol. 9464, International Society for Optics and Photonics, p. 94640D.

Schwarte, A., Haase, P., Hose, K., Schenkel, R. and Schmidt, M. (2011), Fedx: Optimization techniques for federated query processing on linked data, *in* 'International semantic web conference', Springer, pp. 601–616.

Sentinel, C. (2000), URL: https://sentinels.copernicus.eu/web/sentinel/home.

SERVIR (2005), 'Servir global service catalogue', URL: https://www.servirglobal.net/.

Sheth, A. (2015), 'Changing focus on interoperability in information systems: From system, syntax, structure to semantics'.

Shu, Y., Liu, Q. and Taylor, K. (2016), 'Semantic validation of environmental observations data', *Environmental Modelling & Software* **79**, 10–21.

Siddiqa, A., Karim, A. and Gani, A. (2017), 'Big data storage technologies: a survey', *Frontiers of Information Technology & Electronic Engineering* **18**(8), 1040–1070.

Smith, B. and Ceusters, W. (2010), 'Ontological realism: A methodology for coordinated evolution of scientific ontologies', *Applied ontology* **5**(3-4), 139–188.

Smith, B., Kumar, A. and Bittner, T. (2005), 'Basic formal ontology for bioinformatics'.

Smith, J. M., Bernstein, P. A., Dayal, U., Goodman, N., Landers, T., Lin, K. W. and Wong, E. (1981), Multibase: integrating heterogeneous distributed database systems, *in* 'Proceedings of the May 4-7, 1981, national computer conference', pp. 487–499.

Steindl, G. and Kastner, W. (2019), Query performance evaluation of sensor data integration methods for knowledge graphs, *in* '2019 Big Data, Knowledge and Control Systems Engineering (BdKCSE)', IEEE, pp. 1–8.

Strange, R. and Zucchella, A. (2017), 'Industry 4.0, global value chains and international business', *Multinational Business Review* .

Stuckenschmidt, H., Parent, C. and Spaccapietra, S. (2009), *Modular ontologies: concepts, theories and techniques for knowledge modularization*, Vol. 5445, Springer.

Suarez-Figueroa, M. C. and Gómez-Pérez, A. (2008), First attempt towards a standard glossary of ontology engineering terminology, *in* 'Proceedings of the 8th International Conference on Terminology and Knowledge Engineering', pp. 1–16.

Suárez-Figueroa, M. C., Gómez-Pérez, A. and Fernández-López, M. (2012), The neon methodology for ontology engineering, *in* 'Ontology engineering in a networked world', Springer, pp. 9–34.

Sure, Y., Staab, S. and Studer, R. (2009), Ontology engineering methodology, *in* 'Handbook on ontologies', Springer, pp. 135–152.

Sutanta, E., Wardoyo, R., Mustofa, K. and Winarko, E. (2016), 'Survey: Models and prototypes of schema matching.', *International Journal of Electrical & Computer Engineering (2088-8708)* **6**(3).

Theodoratos, D. (2002), Semantic integration and querying of heterogeneous data sources using a hypergraph data model, *in* 'British National Conference on Databases', Springer, pp. 166–182.

Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H. and Hübner, S. (2001), Ontology-based integration of information-a survey of existing approaches., *in* 'Ois@ ijcai'.

Wiederhold, G. (1992), 'Mediators in the architecture of future information systems', *Computer* **25**(3), 38–49.

Xiao, G., Ding, L., Cogrel, B. and Calvanese, D. (2019), 'Virtual knowledge graphs: An overview of systems and use cases', *Data Intelligence* **1**(3), 201–223.

Xu, D. (2017), Contribution to the elaboration of a decision support system based on modular ontologies for ecological labelling, PhD thesis.

Yu, L. (2011), *SPARQL: Querying the Semantic Web*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 241–290.
URL: *https://doi.org/10.1007/978-3-642-15970-1*$_6$

Zhang, F., Zhong, S., Yao, S., Wang, C. and Huang, Q. (2016), 'Ontology-based representation of meteorological disaster system and its application in emergency management', *Kybernetes* .

Zidane, A., El-Bastawissy, A. and Hegazi, O. (2018), 'V-dif: Virtual data integration framework', *Journal of Computer Information Systems* **5**, 26–32.

Abstract — Driven by the need for natural disasters prevention, an increasing number of environmental monitoring systems have been implemented, generating a large amount of data. Nevertheless, the exploitation of this data in integrated studies is still limited, considering siloed sources and data heterogeneity. In this thesis, we proposed a knowledge hypergraph-based approach for data integration and querying. We first developed a modular domain ontology to formalize the knowledge of the environment domain and ensure semantic interoperability among heterogeneous data. Second, we proposed an automatic data integration approach that virtually integrates data in a highly semantic and expressive way using knoweldge hypergraphs. These hypergaphs are then used to enhance query processing in terms of accuracy, completeness, and semantic richness of response. The proposed approach is implemented in an open-source tool and has proved its effectiveness through a real use case.

Keywords: Semantic interoperability, Ontology, data integration, knowledge hypergraph, query processing, environmental monitoring.

Résumé — L'évolution du nombre de données hétérogènes et dispersées en silos, générées par les systèmes d'observation de l'environnement avait entravé leur exploitation dans le cadre d'études intégrées. Dans cette thèse, nous proposons une approche d'intégration et d'interrogation des données multi-sources, basée sur un hypergraphe de connaissances. D'abord, nous avons développé une ontologie modulaire pour formaliser les connaissances liées au domaine de l'environnement que nous avons ensuite exploitée dans l'élaboration d'une approche sémantique d'intégration virtuelle de données fournissant une vue hautement sémantique et expressive à l'aide d'un hypergraphe de connaissances. En tirant profit des hypergraphes, nous avons ensuite amélioré le traitement des requêtes en termes de complétude des réponses et de leurs richesses sémantiques. Notre approche est concrétisée par le développement d'un outil dont l'efficacité a été prouvée moyennant l'évaluation d'un cas réel.

Mots clés : Interopérabilité sémantique, ontologie, intégration des données, hypergraphe des connaissances, Traitement de requêtes, surveillance de l'environnement.

Laboratoire Génie de Production de l'École Nationale d'Ingénieurs de Tarbes
47, avenue d'Azereix - BP 1629 - 65016 Tarbes CEDEX