



**HAL**  
open science

# Knowledge hypergraph-based approach for data integration and querying: Application to Earth Observation

Maroua Masmoudi, Sana Ben Abdallah Ben Lamine, Hajer Baazaoui Zghal,  
Bernard Archimede, Mohamed Hedi Karray

► **To cite this version:**

Maroua Masmoudi, Sana Ben Abdallah Ben Lamine, Hajer Baazaoui Zghal, Bernard Archimede, Mohamed Hedi Karray. Knowledge hypergraph-based approach for data integration and querying: Application to Earth Observation. *Future Generation Computer Systems*, 2021, 115, pp.720-740. 10.1016/j.future.2020.09.029 . hal-04456331

**HAL Id: hal-04456331**

**<https://hal.science/hal-04456331v1>**

Submitted on 15 Feb 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Knowledge hypergraph-based approach for data integration and querying: Application to Earth Observation

Maroua Masmoudi<sup>a,b,\*</sup>, Sana Ben Abdallah Ben Lamine<sup>b</sup>, Hajer Baazaoui Zghal<sup>b</sup>, Bernard Archimede<sup>a</sup> and Mohamed Hedi Karray<sup>a</sup>

<sup>a</sup>INP-ENIT, University of Toulouse, France

<sup>b</sup>Riadi Laboratory, University of Manouba, Tunisie

---

## ARTICLE INFO

### Keywords:

semantic data integration  
Knowledge graph  
ontology  
query processing  
knowledge hypergraph  
earth observation

---

## ABSTRACT

According to the world economic forum report, around 70% of generated data are not used. This limitation of usage is mainly due to the lack of interoperability and linking of data that resides in isolated silos. Indeed, the overwhelming amount of data has worsened heterogeneity problems, as have the types of sources generating data in heterogeneous formats and different semantics. Those data related problematics are frequent in the domain of Earth Observation (EO). Earth observed data use different terminologies, which are difficult to reconcile because they reflect overlapped disciplines. These issues lead to misunderstandings and inefficient exchange and management of data in terms of access, pricing, and data rights, which can hamper environmental phenomena understanding. Virtual Knowledge Graph (VKG), allows semantic integration of existing data sources into a wide Knowledge Graph. In this work we propose a knowledge hypergraph-based approach for data integration and querying, with an application to Earth Observation data. Our proposal takes place in two phases (1) a knowledge hypergraph-based virtual data integration and (2) a hypergraph-based query processing. The first phase allows to generate a virtual knowledge hypergraph consisting of RML mappings between an ontology and the data. The second phase consists of enhancing the user's query by extracting and consolidating a global view of data from different sources based on the generated knowledge hypergraph. The proposed approach is implemented in the Onto-KIT tool (Ontology-based Knowledge hypergraph data Integration and querying Tool) and evaluated through real use case studies. The obtained results show that our proposal enhances query processing in terms of accuracy, completeness, and semantic richness of response. .

---


## 1. Introduction

In recent years, the number of data sources and the amount of generated data are increasing continuously. This voluminous data could be significantly exploited in different domains. However, according to a world economic forum report [28], around 70% of generated data are not used. This limitation of usage is mainly due to the lack of interoperability and linking of data that are in isolated silos. Indeed, the overwhelming amount of data has worsened heterogeneity problems which can appear at different levels. In fact, data are generated in different formats (databases, semi-structured files, images, etc.). Besides, each data source has its own and different data model or schema. Furthermore, data is semantically heterogeneous (synonymy, polysemy, abbreviation, etc.). Each source offers data or semantic models encoding domain knowledge that resides in the experts' minds. Thus, experts or data analysts need to establish contact with original data sources and model producers to understand and use them properly. Undoubtedly, we have not reached a level where data and models are interoperable and linked so that the experts can reuse them soundly [4]. We are still far away from the vision of common information space [16]. Those data related issues are frequent in the domain of Earth Observation (EO). EO data includes hundreds of millions of climate data, ocean data, land data, etc. stored in different

formats (databases, CSV files, Raster images, etc.) and use different terminologies, which are difficult to reconcile because they reflect different disciplines. Our purpose is to break down with those silos and deal with heterogeneity issues that hamper information exchange and interoperability among data sources [23] to provide what we call a global view of information, where different systems and programs will have unhampered and uniform access to the available environmental data that will be linked and synthesized into a single knowledge graph. This global information view allows 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 [14]. To provide a common and unified representation, data needs to be semantically integrated [15]. Data integration is the process of combining data retrieved from multiple and independent sources to provide an integrated and interoperable structure [22]. Currently, several works and approaches are aiming to solve the aforementioned data integration problems, many of which are based on the Semantic Web (SW) technologies [17] [21] [27]. Ontologies are a potential solution for data integration with SW technologies. They capture implicit knowledge across heterogeneous data sources and create a semantic link between them. Semantic data integration approaches can be divided into two main categories; materialized and virtual approaches. Materialized data integration approaches are efficient in terms of query processing since data is gath-

---

\*Corresponding author

 maroua.masmoudi@enit.fr (M. Masmoudi)

ORCID(s):

ered into a single source (usually a data warehouse). However, they are expensive in maintenance and implementation and there is no guarantee that the data loaded into the data warehouse is up-to-date. Therefore, virtual approaches have been proposed to avoid the cost of materialization. These approaches maintain the data in their original sources and access them through an intermediate infrastructure. Virtual Knowledge Graph (VKG), as a multi-relational graph-based paradigm for data integration, could be a suitable solution [32]. It allows experts to retrieve correlated information, find meanings out of its correlations and perform inference over the data and knowledge, and thus derive new implicit knowledge from the explicitly asserted one. Derived knowledge can be used to enrich the answers to queries. However, as the VKG grows in size, the exploitation of data and the answering of users' queries become difficult. 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. Our aim is to create a VKG that enhances the query processing in terms of completeness and relationship richness.

This paper describes a knowledge hypergraph-based data integration and querying approach, with an application to EO data. The aim is to ensure semantic interoperability, the semantically integrate and link the multi-source data in order to guarantee a global information view and to ensure an enhanced information extraction in terms of accuracy, completeness, and relationships richness. According to those challenges, our proposal takes place in two phases (1) the

knowledge hypergraph-based virtual data integration and (2) the hypergraph-based query processing. The first phase semantically links data so as to build a huge knowledge hypergraph that provides a global information view that taking full advantage of heterogeneous data. The second phase proposes 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. 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 hyperERgraph 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. The remainder of this paper is organized as follows: In Section 2, we discuss related work drawing from prior researches in the area of semantic data integration, and we synthesize limitations and drawbacks that we intend to overcome. Section 3 presents background information on the mapping language RML and hypergraphs. Our approach to propose a knowledge hypergraph-based virtual data integration and querying is detailed in Section 4. Section 5 presents the Onto-KIT tool that we evaluate in a real-world use case. Section 6 is where we report on our evaluations of the implementation. Finally, in section 7, we conclude and discuss future directions.

## 2. Related work on semantic data integration approaches

In the literature, various data integration approaches have been proposed. We situate our research in the area of ontology-based data integration approaches (OBDI). To compare the reviewed data integration approaches, and according to the objectives mentioned above, we define comparison criteria following the dimensions (or aspects) of the data integration landscape proposed in [26]. In fact, for surveying data integration approaches, Mountantonakis and Tzitzikas describe an integration process through a multidimensional space. This space is defined by five dimensions: Dataset types, output types, Integration architecture (ETL, mediator, etc.), Internal services (such as schema mapping, semantic vocabulary, query answering), and auxiliary services (e.g. provenance). Accordingly, we highlight six criteria, as shown in Table 1.

- Data acquisition: We identified three mainstream approaches for acquiring data from local sources, namely ETL, OBDA, and mediator.
- 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 (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. Relational databases and geospatial data are the most common data types, as shown in Table 1.
- **Mapping complexity:** reflects the complexity of relations between data sources and the semantic vocabulary (the ontology). This characteristic is important due to the differences in data integration approaches variant 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.

As illustrated in the table, research works related to the area of OBDI generally fall into two primary classes, depending on the type of data access, the virtual, and the materialized integration approaches. The materialized approach is the static transformation of distributed data into a data warehouse or a single RDF store. The whole process is called Extract-Transform-Load (ETL) [31]. 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. Several approaches have been proposed to integrate heterogeneous data into an ontology-based graph such as [27] and [1]. However, these materialized approaches have major disadvantages, including the need of extra storage since data is replicated, this implies an extra cost of storage, as well as the cost of maintenance. 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 (mediator) 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. Indeed, the data remain located in their original sources, and mappings of the data to a semantic vocabulary (for example, an ontology) are generated and results in a virtual RDF graph. Usually wrappers are the components that perform the mapping of queries. They allow the virtual data integration system to access the content of the sources in a uniform language. Based on the mappings, they decompose or rewrite a request in a specific query language that may be processed by the data source. Then, they recombine the partial answers into one only answer in accordance with the virtual RDF graph schema. This approach has the advantage of avoiding the cost of materialization and can profit from more than 30

years' maturity of relational data systems (efficient query answering, security, robust transaction support, confidentiality, etc.). 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.

To overcome the aforementioned limitations of the reviewed data integration approaches, we propose a semantic data integration and querying approach relying 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. Furthermore, to enhance the query response, 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. That's why we need an extra layer to further link the mappings between the data and the ontology in order to extract new knowledge without accessing materialized data. Accordingly, 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.

### 3. Background

In this section, we introduce some basic concepts that we will use in our approach. Specifically, we present RML (RDF mapping language) (Section 2.1) and hypergraphs (Section 2.2).

#### 3.1. RML: RDF Mapping Language

Several mapping languages have been proposed to represent schema mappings such as D2RQ [10], R2RML (RDB to RDF Mapping Language) [11] and RML (RDF mapping Language) [12]. In our work, we use RML since it defines mappings from any data to RDF. In fact, RML extends the mapping language R2RML that maps data from relational databases to RDF, by including mappings of various data formats (XML, JSON, CSV). An RML mapping defines a mapping from any data to RDF. It consists of one or more triples maps. A triples map (tp) is composed of exactly one logical source (property `rml:logical Source`), one subject map (property `rr:subjectMap`) and any number of predicate object maps (property `rr:predicateObject Map`). The RML logical source extends the R2RML logical table and points to

**Table 1**  
Comparison of existing semantic data integration approaches

Approach	Data acquisition	Data access	Semantic Vocabulary	Data types	Mapping complexity	Query processing type
Ontop-spatial [5]	OBDA	Access virtual RDF graphs defined through R2RML mappings.	OGC standard GeoSPARQL	geospatial databases	Ontop-spatial supports two mapping languages: the W3C RDB2RDF Mapping Language (R2RML), and the native Ontop mapping language	Distributed
Geotriples [21]	ETL and OBDA with the stSPARQL/GeoSPARQL evaluator component.	Access on the generated RDF data stores, or querying over virtual RDF graphs defined through R2RML mappings.	OGC GeoSPARQL standard	Geospatial data (shape-files, CSV, XML, and spatially-enabled RDBMS).	Semi-automatic mappings generation. To utilize a different vocabulary, generated mappings may be manually revised.	Centralized and distributed
Optique [20]	OBDA	Access virtual RDF graphs	Siemens ontology	Relational databases, streaming and sensor data	semi-automatically bootstrap an initial ontology and mappings.	Distributed
Ontop [9]	OBDA	Access virtual RDF graphs defined through R2RML mappings.	OGC standard GeoSPARQL	Relational databases.	Supporting two mapping languages: the W3C RDB2RDF Mapping Language (R2RML), and the native Ontop mapping language.	Distributed
[2]	ETL	Querying the global ontology	Creating a local ontology for each data source.	Data formats able to be transformed into a NoSQL database.	Defining rules to generate the corresponding MongoDB database to a data source. Defining a local ontology for each data source, then integrating ontologies in a global one.	Centralized
TripleGeo [27]	ETL	Access on the centralized RDF data repository	OGC GeoSPARQL standard	Geospatial data (shape-files and DBMS)	Manual effort to define the ETL rules.	Centralized
[1]	ELT	Direct access to the (aligned/re-structured) local ontologies, and access to the shared vocabulary, where the system queries each local ontology and merges the results.	The CBROnto ontology	Relational databases, RDF, Spreadsheets.	Automatic generation of the mappings between local and global ontologies using semantic relations.	Centralized
Karma [17]	ETL	Access on the generated RDF data store.	An ontology as input	CSV, JSON, XML, RDF, relational databases	A lot of human intervention in modeling.	Centralized

the data source (property `rml:source`); this may be a file on the local file system or data returned from a Web service. Naming the data source within the mapping makes it possible to map several related data sources simultaneously. A reference formulation (property `rml:reference` Formulation) names the syntax used to reference data elements within the logical source. As of today, possible values are `ql: JSON-Path`, `ql: XPath`, `ql: CSS3`, and `rr: SQL2008`. The subject map specifies how to define the subject of each tp and its optional type of URI. A predicate-object map consists of pairs of predicate maps (property `rr:predicateMap`) and object maps (property `rr:objectMap`) that, together with the subjects generated by the subject map, may form one or more RDF triples for each row/record of the (database/CSV/XML/JSON source respectively). 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`. 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.

### 3.2. Hypergraphs

A hypergraph is the generalization of an ordinary graph by defining edges between multiple vertices instead of only two vertices. In what follows, we provide a set of definitions presented by [7].

**Definition 1.** A hypergraph  $H$  is a pair  $\langle V, E \rangle$  where:

- (i)  $V = v_1 \dots, v_n$  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 2.** A subhypergraph  $H(V')$  of the hypergraph  $H$  is the pair  $\langle V', E' \rangle$  where :

- (i)  $V' \subset V$ ,
- (ii)  $E' = (e_j)_j \in J$  such that for all  $e_j \in E' : e_j \subseteq V'$  and  $J \subseteq I$

The notion of hypergraph may be generalized in a way that the hyperedges can be represented in certain cases as vertices, i.e. a hyperedge  $e$  may consist of both vertices and hyperedges as well. For example:

**Definition 3.** A Generalized hypergraph  $GH = \langle V, E \rangle$  where:

- (i) Let  $V = v_1; v_2; v_3$
- (ii)  $E = e_1 = v_1; v_2; e_2 = v_2; v_3, e_1; e_3 = v_1; e_1; e_2$ .

**Definition 4.** Directed hyperedge (hyperarc)  $\vec{e}_i$  is an ordered pair  $\vec{e}_i = (e_i^+, i); \vec{e}_i^- = (i, e_i^-)$ ; 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  $\vec{e}_i$ .

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

## 4. Knowledge hypergraph-based data integration approach

### 4.1. Motivating example

To extract the important problematics and identify the objectives of our approach, we define a running example to which we shall refer all along with this paper. We consider three different data sets about precipitation from three different data sources; A raster image in ENVI<sup>1</sup> format which is a flat-binary raster file provided by the Observatory of Sahara and Sahel (OSS<sup>2</sup>), JSON data extracted through the AerisWeather API<sup>3</sup> and a CSV dataset provided by the National Oceanic and Atmospheric Administration (NOAA<sup>4</sup>). This multi-source data presents heterogeneous data schema (metadata), depicted in Figures 1-2. We noted the following

```
ENVI
description = Climate Hazards Group InfraRed Precipitation
with Station data (http://chg.geog.ucsb.edu/data/chirps/)
samples = 7200
lines = 2000
...
band names = {rainfall}
map info = {Geographic Lat/Lon, 1, 1, -180, 50, 0.050000000000000,
0.050000000000000, WGS-84, units=Degrees}
values = {rainfall, mm, 0, 10000, 0, 10000, 0, 1}
...
date = 20191016
sensor type = CHIRPS
```




Figure 1: Partial view of the metadata from the raster image originating from the OSS.

```
[{"loc":
{"lat":44.9778,
"long":93.265},
"periods": [{"summary": {
"timestamp":1571202000,
"dateTimeISO":"2019-10-16T00:00:00-05:00",
"range": {"fromTimestamp":1571202000,
"toDateTimeISO":"2019-10-16T00:00:00-05:00",
"timestamp":1571288395,
"dateTimeISO":"2019-10-16T23:59:59-05:00",
"count":0},
"precipitation": {
"totalMM":8.88,
"totalIN":0.35,
"count":12}}}], -1 ]
```

**a**

<b>STATION</b>	<b>CITY</b>	<b>DATE</b>	<b>HCPC</b>
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

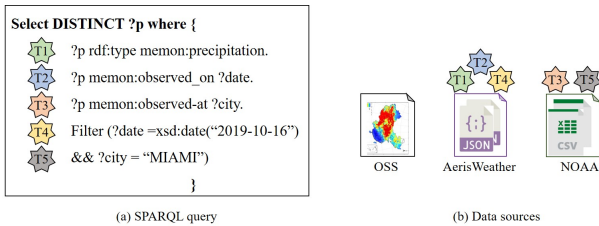
**b**

Figure 2: (a) Partial view of data about precipitation from the AerisWeather API. (b) Partial view of data about Hourly Precipitation (HCPC) from the NOAA. Data schema in bold.

concerns and issues. First, various terms are used to describe the same real-world feature that refers to "precipitation" despite they correspond to the same meaning. For instance,

<sup>1</sup><https://www.harrisgeospatial.com/Software-Technology/ENVI>  
<sup>2</sup><http://www.oss-online.org/>  
<sup>3</sup><https://www.aerisweather.com/>  
<sup>4</sup><https://www.noaa.gov/>

OSS uses the word “rainfall” to refer to “precipitation” in AerisWeather, whereas NOAA uses the term “HPCP”. The variety of terms complicates the work of experts and software agents who 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 lists the precipitation values within a city described by the name. Finally, the date of observation is differently represented in the three datasets: “yyymmjj” format for the first dataset, “yyyy-mm-jjThh:mm:ss” format for the second dataset and “yyymmjj hh:mm.” for the third one. Let us now consider an example of a SPARQL query based on the MEMOn ontology (Modular Environmental Monitoring Ontology) [25], illustrated in Figure 3. The query asks for precipitation data observed in “Miami” on 16 October 2019 and comprises five triple patterns T1, T2, T3, T4, and T5. In this example, the AerisWeather API can answer T1, T2, and T4. T3 and T5 can be answered on NOAA while OSS cannot answer any triple pattern since the OSS vocabulary is different from the SPARQL quer’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



**Figure 3:** Motivating example. (a) SPARQL query over data sources; (b) heterogeneous data sources.

in the three schemas: by the term "rainfall" in the OSS, the term "precipitation" in the AerisWeather API and the term "HPCP" 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 "HPCP" and "precipitation", it cannot resolve the difference between the two spatial representations (Lat/Long and country) or the difference between the heterogeneous formats of the date which lies a big issue. Thus, the query processing will not guarantee complete results from the different data sources. In other words, a traditional query processing cannot ensure finding all results because the data sources contain different data schema. For those reasons and to achieve our goals, we proposed a novel knowledge hypergraph-based data integration and querying approach

that (1) virtually integrates EO data by maintaining it in their sources in order to have a global knowledge hypergraph and (2) provides a query processing approach that offers an optimal result for SPARQL queries.

#### 4.2. The architecture of the proposed approach

The use of architecture and design patterns have impacts on the quality attributes of a system. In this work, four quality attributes are aimed to be fulfilled which are interoperability, completeness, usability and maintainability. Bi et al, [6] and Harrison et al, [19] analyzed the impact of architectural designs on several quality attributes. Based on their works, we choose the layered architecture that presents positive impacts on maintainability. Considering these quality attributes and according to the needs highlighted in the motivating example, the layered architecture of the proposed approach is depicted in Figure 4. It is composed of four tiers:

- (1) **The data layer** encompasses different data sources relevant to earth observations and deals with different data formats (i.e.: CSV, RDB, JSON, etc.).
- (2) **The semantic layer** consists of 3 components, i.e., the Modular Environmental Monitoring Ontology (MEMOn), the spatial and the temporal RDF stores named  $S_{RDFStore}$  and  $T_{RDFStore}$ , respectively. MEMOn was proposed as a modular ontology for the environmental monitoring field, based on the upper-level ontology Basic Formal Ontology (BFO) [3] and other existing ontologies such as the Common Core Ontologies (CCO) [30], the Semantic Sensor Network ontology (SSN) [18] and the Environment Ontology (ENVO) [8]. The modules of MEMOn incorporate all the different kinds of information entities handling all contexts of environmental phenomena, e.g., flooding, earthquakes, etc., spatial and temporal information, and sensing and observation information that are of importance to the environmental monitoring domain. And the links between modules cover the 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 the EO data. They aim to ensure the schematic and semantic interoperability between the different spatial (resp., temporal) representations extracted from the heterogeneous data sources. They are adopted, in our approach, since they are aligned to MEMOn ontology. This layer aims to ensure semantic interoperability.
- (3) **The data integration layer** aims to ensure other quality attributes such as structural, schematic interoperability and completeness of the query response. It 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 OBDI paradigm. It comprises three steps: (a) semantic annotation, (b) RML mappings

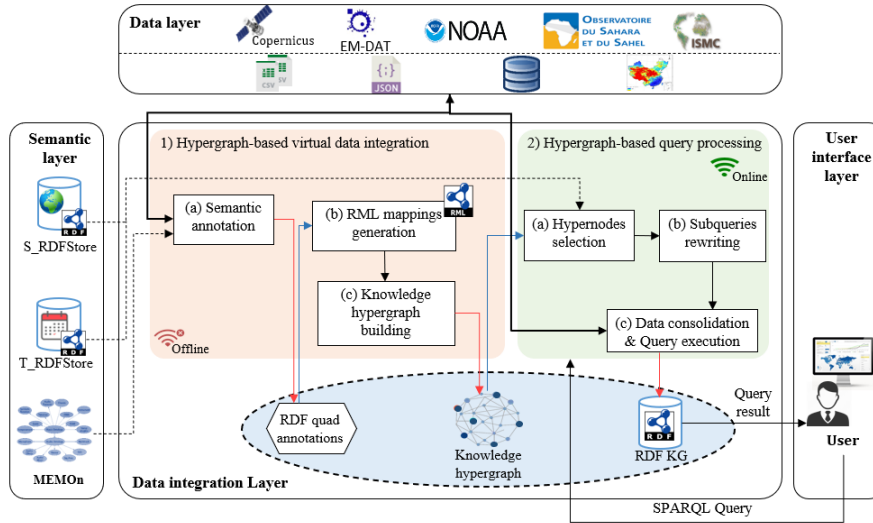


Figure 4: The architecture of the proposed knowledge hypergraph-based approach for data integration and querying.

generation, and (c) knowledge hypergraph building, as shown in Figure 4. 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 comprises three main 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. This layer will be discussed in detail throughout the paper.

- (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. This layer will help fulfill usability which is concerned with the ease with which a user can complete tasks.

### 4.3. 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, represents the ontology. Its goal is to provide a formal and high-level representation of the domain of interest.
- The data layer, represents the available data and its metadata.
- And 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. They are used to translate the query on

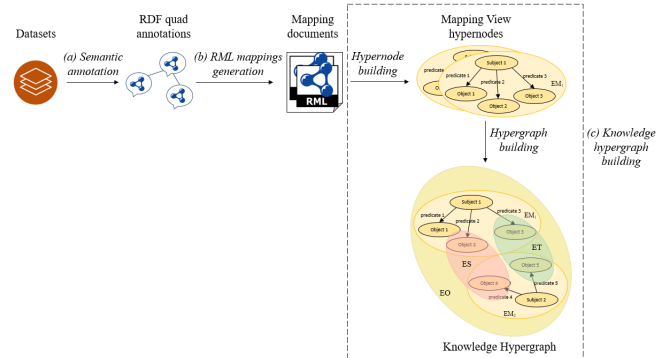


Figure 5: The whole process of the hypergraph-based virtual data integration.

the ontology into a query that may be processed by the data source.

The whole process of this phase is illustrated in Figure 5. 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 RML mapping view hypernodes and various hyperedges to semantically describe different views of the environmental observations.

#### 4.3.1. 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) strat-



egy, 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 [29]. 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. This process is done by annotating the source metadata to allow semantic interoperability. Based on the SE strategy, we define an annotation as follows (Definition 5):

**Definition 5.** *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  $A = \langle O, C, T, S \rangle$  has the following components:*

- $O$  = some ontology class,
- $T$  = data term,
- $C$  = some relation between  $O$  and  $T$  (the relation will be rdfs: label),
- $S$  = a reference to the source from which the data term is taken.

Hence, to represent annotations, we choose to use the RDF Quad statements, also called Named Graphs<sup>5</sup>. 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 5 so that annotations with provenance information can be represented in definition 6:

**Definition 6.** *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*

$$\langle O, C, T, S \rangle \in (R \cup B) \times R \times (R \cup B \cup L) \times (R \cup B),$$

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 using a domain ontology. Initially, the system extracts the entities from the metadata (M) (line8). 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.

<sup>5</sup><https://www.w3.org/2009/07/NamedGraph.html#named-graphs>

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<sup>6</sup> and its filtering that aims to limit the extraction of other information that may be non-useful to the integration process. 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 this end, each entity of the metadata ( $m_i$ ) is mapped to one class from the ontology ( $c$ ) (line 11). If no matching is identified between  $m_i$  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 are matched 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, the system generates the annotation  $A$  comprising the metadata entity  $m_i$ , the class  $c$ , and the data source  $C_o$  (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 quads ( $Set_A$ ), that semantically annotate the data. The semantic annotation process goes beyond the exploitation of the ontology and the thesaurus, as it allows the exploitation of the generated  $Set_A$ . The core idea is to exploit the knowledge inferred from the domain ontology and the thesaurus, represented in the  $Set_A$  to identify plausible semantic annotations for a new source metadata.

#### 4.3.2. RML mappings generation

The proposed system 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 ontology and the annotations produced in the previous step. 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). To generate RML mappings, we introduce the following substeps:

1. For each metadata, we create a new triples map.
2. For each triples map, we generate a subject map that defines the rule that generates 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.

<sup>6</sup><http://www.gdal.org/>

**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 :**

$M = \{m_1, m_2 \dots m_n\}$  : 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

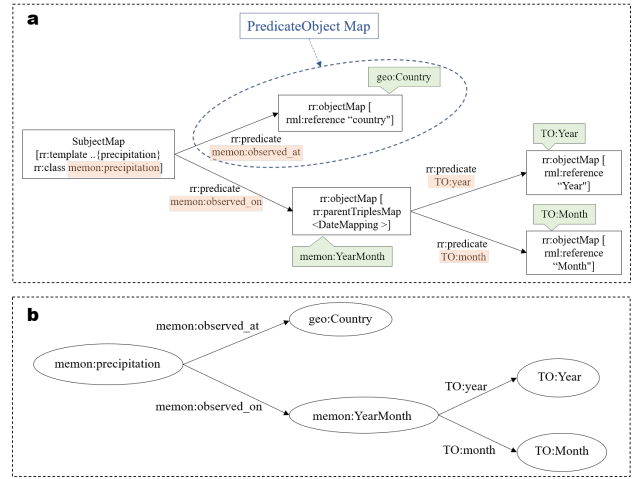
**4 begin**

```

5   initialization;
6    $P \leftarrow$  rdfs:label;
7    $Co \leftarrow$  source;
8    $M \leftarrow$  extractMetadata(data);
9   foreach metadata item  $m_i \in M$  do
10    if Match ( $m_i, Set_A$ ) then
11       $c \leftarrow$  find ( $m_i, Set_A$ );
12    end
13    else if Match ( $m_i, C$ ) then
14       $c \leftarrow$  find ( $m_i, C$ );
15    else
16       $Set_T \leftarrow$  find ( $m_i, T$ );
17       $c \leftarrow$  find ( $Set_T, C$ );
18    end
19     $S \leftarrow c$ ;
20     $O \leftarrow m_i$ ;
21     $A =$  createQuad ( $S, P, O, Co$ );
22     $Set_A \leftarrow Set_A + A$ ;
23  end
24 end

```

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.



**Figure 6:** (a) An RML mapping graph (RML\_G). (b) A semantic mapping view over RML\_G

**4.3.3. 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 used not just as 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 7.

**Definition 7.** Formally, an RML mapping graph is denoted as  $RML\_G = \langle V, E \rangle$  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, as shown in figure 5: hypernodes building and hypergraph building and/or extending<sup>7</sup>. 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

<sup>7</sup>When a new data source is added.

modeled as edges. We illustrate in Figure 6(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 6(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 is hypernode defined as follows (Definition 8).

**Definition 8.** (A hypernode) Given a hypergraph  $\langle V, E \rangle$ , a hypernode<sup>8</sup> is mainly defined as a set of nodes  $U \subseteq V$  that act together as a single unit.

(A mapping view hypernode) 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 a knowledge graph [14], we define a knowledge hypergraph in definition 9.

**Definition 9.** 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 10.

**Definition 10.** The *knowledge Hypergraph* is a generalized hypergraph with directed and undirected hyperedges. It can be designated as the tuple:

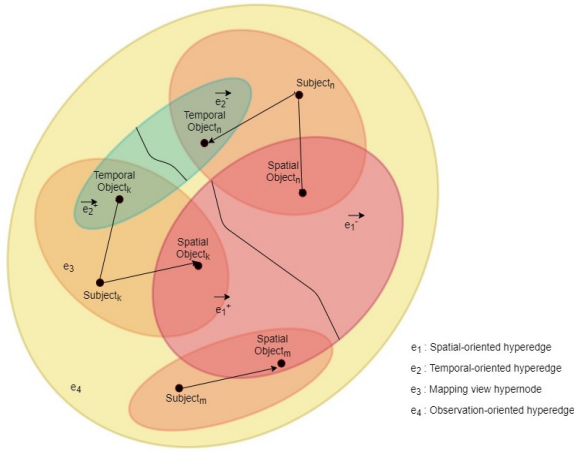
$$\langle V, A, E, E_D, E_M, E_O, \lambda_{label}, \lambda_v \rangle, \text{ where:}$$

<sup>8</sup>Hypernodes may be referred to as undirected hyperedges, compound nodes, or metanodes in the literature.

- $V = V_s \cup V_o$  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  $\langle i, j \rangle$ , where  $i, j \in V$ .
- $E$  is the set of hyperedges.  $E = E_D \cup E_O$
- $E_D = E_S \cup E_T$  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  $\subseteq V_o$  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  $\subseteq V_o$  and refer to the objects describing the temporal context of the observation.
- $E_M$  consists of the mapping views represented as hypernodes. Each  $e_m \in E_M$  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  $e_o \in E_O$  embodies the hypernodes sharing the same subject.
- $\lambda_{label} : E \mapsto S$  is a hyperedge-labeling function. Given a hyperedge  $e \in E$ , its hyperedge label is deduced according to the pattern of the hyperedge.  $S$  is the set of labels.
- $\lambda_v : V^2 \mapsto R$  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  $E_o$  also represents an observation-oriented sub-hypergraph, described by definition 11.

**Definition 11.** The *observation-oriented Sub-hypergraph* consists of:

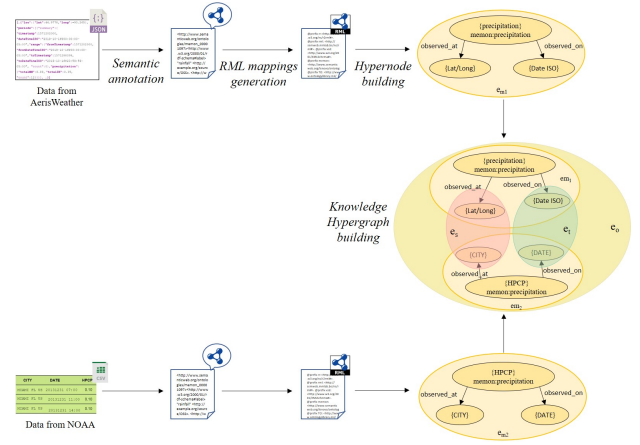


**Figure 7:** Observation-oriented Sub-hypergraph describing the relations among observations.

1. A finite set of vertices  $U \subseteq V$ .
2. A finite set of hypernodes,  $E_M = e_{m_1}, \dots, e_{m_n}$ ; 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 7 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 map plays the input or output role in a function  $\lambda_v$ . An observation-oriented hyperedge ( $e_4$ ) can be defined as a collection of the mapping view hypernodes whose subjects share the same ontological class.

To recapitulate, Figure 8 shows an example to explain the process of the hypergraph-based virtual integration, starting with the semantic annotation of two different formats of EO data (a CSV and JSON 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 8:** The process of the virtual integration from data to knowledge hypergraph building.

#### 4.4. Hypergraph-based query processing

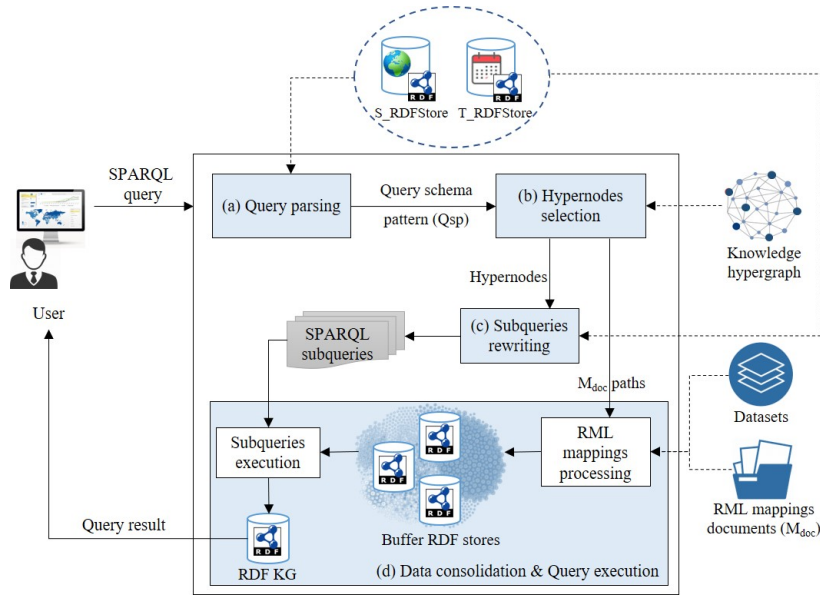
The hypergraph-based query processing phase allows the extraction and the consolidation of data into RDF format through answering SPARQL queries. The whole process takes as input a SPARQL query. Based on the knowledge hypergraph, resulting from the first phase, it generates RDF triples as a result of 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 global view answer over all data sources, we introduce a four-step approach:

- (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 9.

##### 4.4.1. 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



**Figure 9:** The entire hypergraph-based query processing.

the schema graph pattern (**SGP**) corresponding to the extracted GP using the domain ontology. In this context, the SPARQL query and the schema graph pattern are defined in Definitions 12 and 13.

**Definition 12.** A SPARQL query is a 4-tuple  $\langle GP, DS, SM, R \rangle$ , 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 [33].

**Definition 13.** (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.

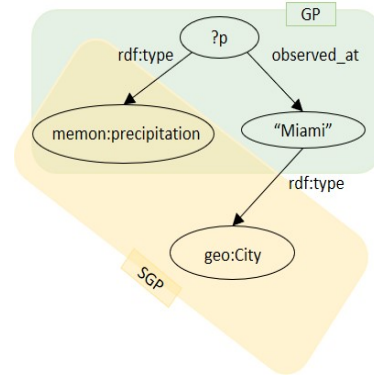
If  $?x \in \text{var}(GP)$  then  $\text{SchemaElement}(?x) \in (SGP)$ .

To illustrate the transformation process of an input QR to SGP, we consider the query in Listing 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 1: A example of SPARQL query.

Figure 10 shows the GP of the query above in green and the corresponding SGP in yellow.



**Figure 10:** The corresponding graph pattern and schema pattern to the query in Listing 1.

In order to extract the schema elements corresponding to the instances of GP, we 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 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 ap-

proach. For example, for the query in Figure 3, the term "Miami" is automatically identified as an instance of the "city" ontological class, and '2019-10-16' 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., '2019-10-16') 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 "2019-10-16" is enriched by new relations from the  $T\_RDFStore$  such as: "2019-10-16" year "2019", "2019-10-16" sameAs "2019/10/16", "2019-10-16" sameAs "16 October 2019", etc. Accordingly, the system translates the initial query to sub-queries according to the different temporal and spatial representations.

#### 4.4.2. 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 14, in which we elaborate on how to figure out which hypernodes are likely to generate RDF triples matching the SGP.

**Definition 14.** *Matching of a 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  $\langle V, A, E, E_D, E_M, E_O, \lambda_{label}, \lambda_v \rangle$

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:

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

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  $TM.obj, TQ.obj \in V \wedge TM.obj \in e \wedge e \in E_D \wedge (\exists v \in V \cap e \cap TM_2) \wedge same(TM.sub, TM_2.sub)$

#### 4.4.3. 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 2 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 ( $\lambda_v$ ). 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.

#### 4.4.4. 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. As a mapping process executor, we opted for RML Mapper<sup>9</sup>, 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. 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,

<sup>9</sup><https://github.com/RMLio/rmlmapper-java>

---

**Algorithm 2:** Translation of Input query to sub-queries

---

```

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

```

---

Algorithm 3 illustrates how the different steps are orchestrated, from the extraction of the GP until the generation of the RDF triples that match it.

## 5. Onto-KIT Tool: implementation and evaluation

The proposed approach is implemented through the tool, named Onto-KIT (Ontology-based Knowledge hypergraph data Integration and querying Tool) and 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 automatically integrates

---

**Algorithm 3:** Knowledge hypergraph querying process

---

```

1 Input :
    $QR$  : SPARQL query
    $K - Hyper$ : The knowledge hypergraph
2 Output :
    $Set_{RDFTriples}$  : Set of RDF triples
3 Variables :
    $Qsp$  : Schema graph pattern
    $HQR$  : Generated query to interrogate K-Hyper
    $Set_{spatial}$ : Extracted spatial representations
    $Set_{temporal}$ : Extracted temporal representations
4 begin
5    $Qsp = extractSchemaGraphPattern(QR);$ 
6   foreach triple pattern  $tp \in Qsp$  do
7     if  $(tp.predicate \in spatialPredicatesSet)$ 
8       then
9          $Set_{spatial} =$ 
10           $ExtractSpatialRepresentations$ 
11           $(Qsp.obj, S_{RDFStore});$ 
12        end
13        if  $(tp.predicate \in temporalPredicatesSet)$ 
14          then
15             $Set_{temporal} =$ 
16              $ExtractTemporalRepresentations$ 
17              $(Qsp.obj, T_{RDFStore});$ 
18          end
19        end
20       $HQR = WriteQuery(Qsp, Set_{spatial},$ 
21         $Set_{temporal});$ 
22       $Map < Set_{EM}, paths > =$ 
23         $QueryK-Hyper(HQR);$ 
24       $Set_{subQR} = RewriteSubQueries(QR, Set_{EM});$ 
25      foreach  $path \in paths$  do
26         $store = RMLMappingProcessing(path);$ 
27         $stores = stores + store;$ 
28      end
29       $ExecuteSubQueries(stores, Set_{subQR});$ 
30    end
31  end

```

---

various formats of data in order to avoid human intervention as much as possible, especially in terms of data mapping and modelling. Specifically, the semantic annotation and mapping generation processes are automatically performed. In addition, our tool performs a novel knowledge hypergraph-based query processing through an enhanced source selection technique. Onto-KIT is implemented in Java and integrates several open-source libraries:

- OWL API<sup>10</sup> : to manipulate the ontology.
- RDF4J<sup>11</sup> : a widely used Java framework for developing Semantic Web applications, tools, and servers.

<sup>10</sup><http://owlapi.sourceforge.net/>

<sup>11</sup><https://rdf4j.eclipse.org/>

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: An open-source client application built on Java, used to create the tool interfaces.

Our implementation is freely available on GitHub<sup>12</sup>. We tested Onto-KIT tool over 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 in terms of completeness. The use case study is based on the example presented in Section 4.1; dataset about floods in CSV format from EM-DAT<sup>13</sup> and heterogeneous precipitation data from three different sources:

- Data in ENVI format (The ENVI image format is a flat-binary raster file) from OSS,
- Data in JSON format from AerisWeather API,
- Data in CSV format from NOAA.

### 5.1. DISERTO Tool

For the virtual data integration phase, DISERTO is implemented to automatically generate the knowledge hypergraph [24]. It is based on the RML model<sup>14</sup>. 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. The outputs consist of the RDF annotations document, the RML mapping documents, and the knowledge hypergraph stored in an RDF4J triple store. As a first step, we selected MEMOn ontology, the UNESCO thesaurus<sup>15</sup>, and the datasets, mentioned before. The system loads the ontology and executes the entire process, schematized by the activity diagram, illustrated in Figure 11. For each data format, an appropriate wrapper is used to read the data and extract the metadata entities  $M = m_{i=1..n}$  (Step 1). For instance, in the case of CSV file presented in the motivating example, the CSV wrapper extracts the columns’ names, which correspond to the metadata entities  $M = (\text{station, city, date, HPCP})$ .

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 searched the corresponding  $c_i$  to “ $m_1 = \text{rainfall}$ ”. 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 JSON data, “ $m_1 = \text{precipitation}$ ” is found in MEMOn as a label to the class `memon_00001097`. Listing 2 presents the two annotations generated for these two metadata entities (Step 6).

```
<http://www.semanticweb.org/ontologies/2017/10/memon_00001097>
<http://www.w3.org/2000/01/rdf-schema\#label> "rainfall"
<http://example.org/source/OSS>.

<http://www.semanticweb.org/ontologies/2017/10/memon_00001097>
<http://www.w3.org/2000/01/rdf-schema\#label> "precipitation"
<http://example.org/source/Aeris>.
```

Listing 2: 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 that defines 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 objects from 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 3 illustrates the resulted RML mapping document for the JSON data as automatically generated by DISERTO.

<sup>12</sup><https://github.com/marouamasmoudi/Onto-KIT>

<sup>13</sup><https://www.emdat.be/>

<sup>14</sup><https://github.com/RMLio/RML-Model>

<sup>15</sup><http://vocabularies.unesco.org/browser/thesaurus/>



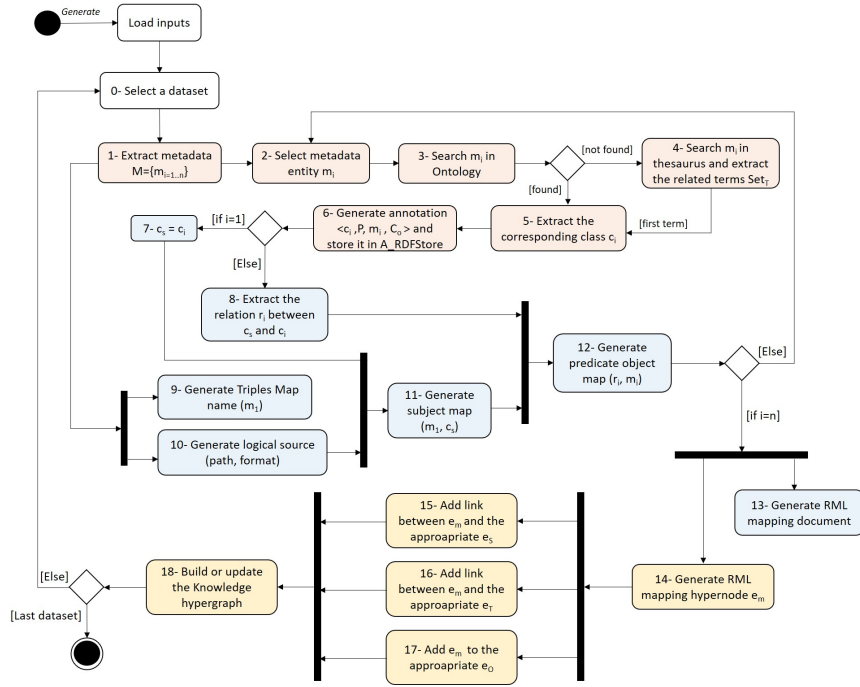


Figure 11: Activity diagram of DISERTO.

```

@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/MidTimeOntology#>
<#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 3: The resulted RML mapping document for the input Precipitation.json from AerisWeather API.

After generating RML mapping documents, DISERTO builds the virtual knowledge hypergraph and stores it into an RDF4J store. To do so, for each RML mapping document, the system 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 obser-

vation hyperedge ( $e_o$ ) (Step 17). Figure 12 shows an illus-

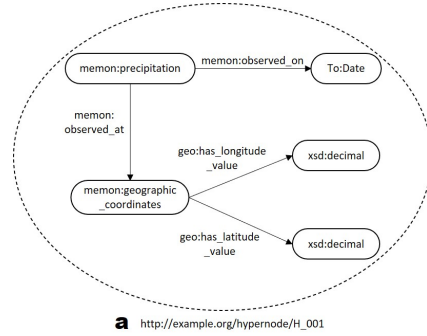


Figure 12: The mapping view hypernode "H\_001".

tration of the mapping view hypernode corresponding to the RML mapping document in Listing 3. The IRI corresponding to this hypernode is "http://example.org/hypernode/H\_001". Figure 13 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".

```

(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_0001]

(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_00001056)
[http://example.org/hypernode/H_0001]

```

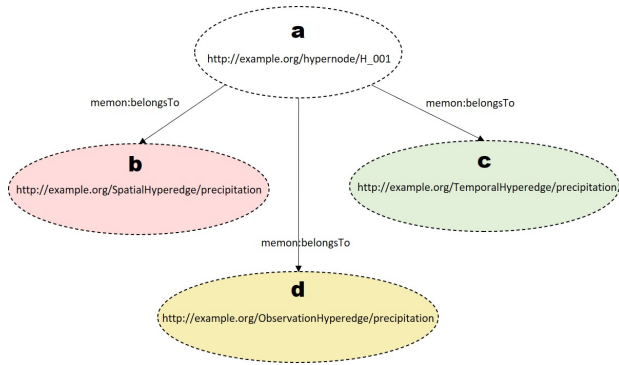


Figure 13: The relations between "H\_001" and corresponding hyperedges.

```
(http://www.semanticweb.org/lenovo/ontologies/2017/10/memon_00001056, http://www.semanticweb.org/lenovo/ontologies/2017/10/belongsTo, http://example.org/hypernode/H_0001)
[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_0001)
[http://example.org/TemporalHyperedge/memon_00001097]

(http://example.org/hypernode/H_0001, http://example.org/path, "D:\DISERTO\Resources\RML_mappings\H_0001.ttl")
[http://example.org/ObservationHyperedge/memon_00001097]
```

Listing 4: Examples of RDF quads representing a partial view of the knowledge hypergraph.

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 4 represents the generated RDF quads that correspond to Figure 13. Figure 14 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 hyperedge

"[http://example.org/SpatialHyperedge/memon\\_00001097](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](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://example.org/ObservationHyperedge/memon_00001097)".

Consequently, the knowledge hypergraph generated in the case of our use case study is illustrated in Figure 15. It contains 2 observation hyperedges (flood and precipitation), 2 spatial hyperedges, 2 temporal hyperedges, and 4 hypernodes (one for each input dataset). Figure 15 also shows the relations among "memon:flood" and "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"

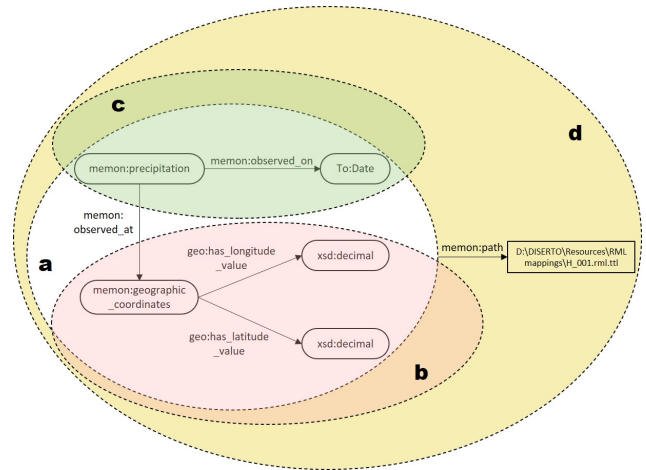


Figure 14: A partial view of the precipitation-oriented sub-hypergraph.

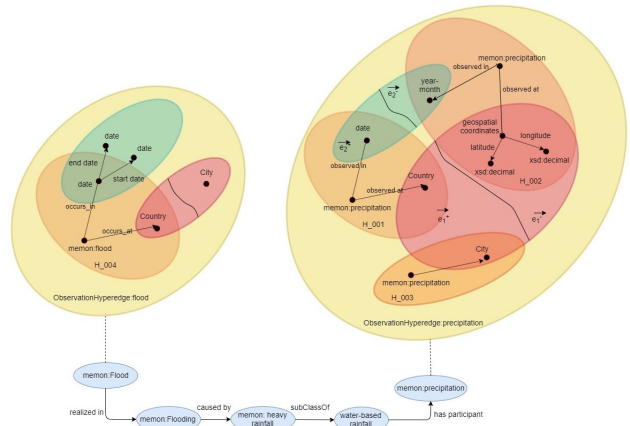


Figure 15: A partial view of the knowledge hypergraph.

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.

### 5.2. 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, (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 illus-

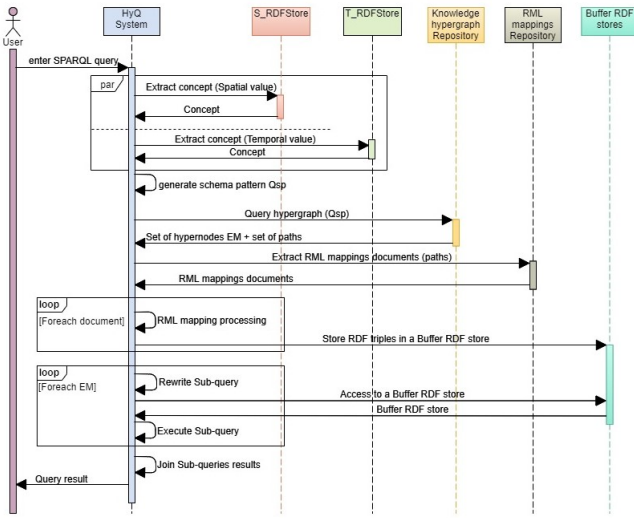


Figure 16: Sequence diagram of the query processing.

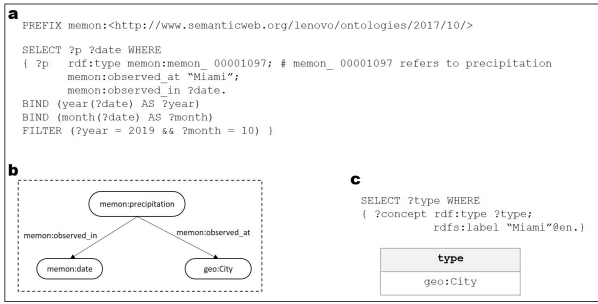


Figure 17: (a) Example of SPARQL query. (b) The corresponding schema pattern. (c) The query used to interrogate  $S_{RDFStore}$

trated through the sequence diagram (cf. Figure 16). Based on the datasets involved in the use case study, we define the SPARQL query defined in Figure 17(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 17.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 17.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 relevant 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

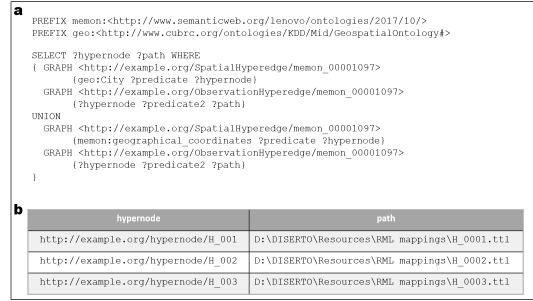


Figure 18: (a) The query used to interrogate the knowledge hypergraph. (b) Results

in the input query. Accordingly, 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 18 (a) illustrates only the part of the query that concerns the spatial context. Figure 18 (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 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: 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 6 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/29mm> rdf:type memon:memon_00001097.
<http://example.com/precipitation/29mm> memon:observed_at dbr:Miami.
<http://example.com/precipitation/29mm> memon:observed_on
<http://example.com/yearmonth/2018,12>.
<http://example.com/yearmonth/2018,12> to:year "2018"^^xsd:int.
<http://example.com/yearmonth/2018,12> to:month "12"^^xsd:int.
```

Listing 6: Examples of generated RDF triples.

## 6. Evaluation and discussion

According to the quality attributes mentioned in Section 4 (interoperability, completeness, usability and maintainability), the implementation of the two independent software modules (DISERTO and HyQ) promotes the maintainability. In addition, the proposed user interface of OntoKIT helps fulfill usability. This section focuses on the two other quality attributes; interoperability and completeness. Indeed, the objectives of our performance evaluation is threefold: first, we aim at evaluating the accuracy of the semantic annotation step and consequently evaluating the accuracy of the generated RML mappings. Second, we aim at evaluating the efficiency of the query processing when using the knowledge hypergraph. Third, we wish to prove that the query processing based on the knowledge hypergraph actually enhances the query's results in terms of completeness and relationship richness.

**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 :

The quality of semantic 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 1), recall measures how many of the actual pairs have been matched (cf. equation 2) [13]. However, we can have excellent precision with terrible recall, or alternately, terrible precision with excellent recall. Thus, we also consider the metric F1-measure which is the average of precision and recall (cf. equation 3). We calculated the three metrics using the equations below:

$$Precision = \frac{TP}{FP \cup TP} \quad (1)$$

$$Recall = \frac{TP}{FN \cup TP} \quad (2)$$

$$F1 - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

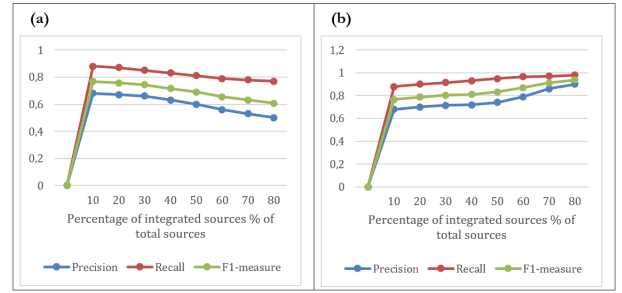
With,

- TP: True Positives is the correctly 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) Execution Time:** Elapsed time between the submission of a query to the system and the delivery of the answers.

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

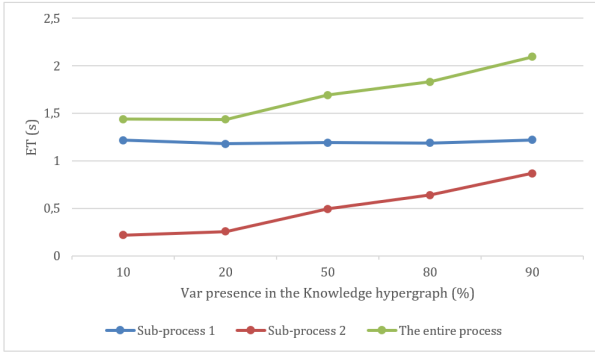
**(iv) Completeness:** Query result percentage with respect to the answers produced by a SPARQL query over all the datasets.



**Figure 19:** Performance variation in terms of precision, recall and F1-measure (a) only with domain ontology exploitation. (b) with ontology, thesaurus and Set<sub>A</sub> exploitation.

### 6.1. Experiment 1: DISERTO; Schema annotation evaluation

Figure 19 shows the performance variation in terms of precision, recall and F1-measure of DISERTO while increasing the number of integrated data sources. Figure 19(a) illustrates the experimental measures of the semantic annotation step only with the exploitation of the domain ontology. In the second experiment, illustrated in Figure 19(b), the system exploits the thesaurus UNESCO and the set of generated RDF quad annotations Set<sub>A</sub> 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 annotation contains the key concepts without useless terms (noise) and that it covers well the modeled domain. It can be seen in Figure 19(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 19(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<sub>A</sub> 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<sub>A</sub> that it stores to be exploited directly next time. The graph in Figure 19(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.



**Figure 20:** Performance variation of HyQ in terms of execution time.

## 6.2. 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 answering 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 3). According to Definition 13, "var" refers to the variable "precipitation" in our input SPARQL query. Figure 20 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. The observed results show that the execution time (ET) of sub-process 1 is almost constant despite the increase in the var presence percentage in the hypergraph. However, the ET of sub-process 2 rises with the rise of the number of hypernodes related to the SPARQL query's var in the knowledge hypergraph. 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 process's ET.

## 6.3. Experiment 3: Comparison with Karma approach

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 [17]. Karma provides a graphical user interface through which we import data. The output from the interaction with the Karma Web system is a materialized RDF data store used for the query-

ing. We executed 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 below, 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 }

```

*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 "yyymmdd" format. In our approach, we used the *T\_RDFStore* to link the different formats of date values of an individual of "bfo:date". The SPARQL query used in this context is presented below. Q2:

```
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)
}

```

*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 used a SPARQL query Q3 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" }

```

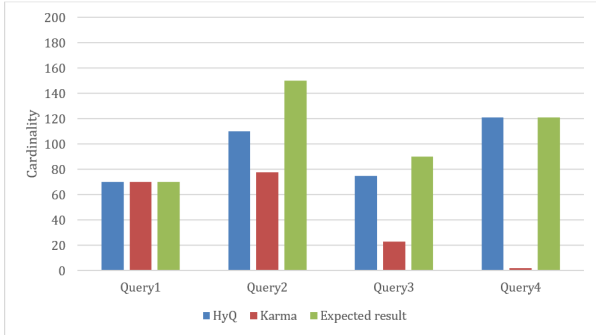
*Type 4: Queries that generate results based on the knowledge encoded in the ontology*

After integrating the precipitation data from the different sources, a SPARQL query, as shown below, 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 SWRL rules existing in the ontology MEMOn 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.

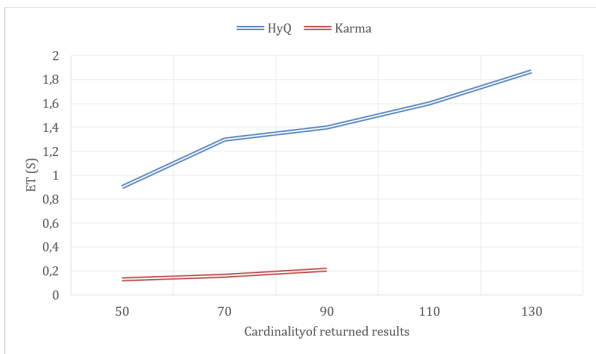
```

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_precipitation;
memon:observed_at ?city }

```



**Figure 21:** Performance variation in terms of completeness and cardinality of returned results.



**Figure 22:** Performance variation in terms of execution time and cardinality of returned results.

Figure 21 reports on completeness and cardinality of returned results for the four queries for both Karma and our approach, whereas Figure 22 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 ( $> 0.8$  s). 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.

## 6.4. Discussion

Onto-KIT aims at effectively addressing semantic data integration and query processing, based on a knowledge hypergraph, while taking into account the results accuracy, completeness and semantic richness. 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 criteria of a data integration approach.

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 no link exists between two sources that may contribute to the final result. 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 the existing VoID (Vocabulary of Interlinked Datasets) technique 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 char-

acteristics and can be even made richer by incorporating other relations such as the causality knowledge between observations and environmental processes, for example.

However, the proposed data integration and querying approach 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. 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. 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

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 other domains with some adaptations by choosing the appropriate ontology and thesaurus.

## 7. Conclusions and future works

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 and relationship. The approach was implemented through Onto-KIT tool that comprises DISERTO and HyQ software modules and works in two phases. In the offline phase, DISERTO automatically generates the RML mappings that can be used to transform the input data into RDF and generates the virtual knowledge hypergraph. In the online phase, HyQ executes an enhanced query processing by improving the source selection task to identify relevant sources that possibly contribute to the final result. Specifically, HyQ executes a SPARQL query and generates an RDF knowledge graph on the basis of the virtual knowledge hypergraph.

This approach is significantly improving the query answer in terms of completeness by using the hypergraphs as a mathematical structure in the virtual KG. Using a hypergraph-based representation to model data has an important benefit of expressing relationships. The main benefit of the proposed 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 GeoSPARQL and Linkededata 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 which specifies the semantics of the data and links multi-source data. Nevertheless, there are still some aspects to be considered. 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, we will integrate additional data formats such as XML and Shape files. In addition, 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 the hypergraph-based learning methods and algorithms, discovering interactions between observations and generating alerts will be easier in the near future. Furthermore, in this work, we focused on the improvement of the query processing in terms of query result completeness. In future works, we intend to use known SPARQL query federations systems and rely on other approaches and methods that could improve the runtime of the query processing in order to ensure the balancing between query answer completeness and query execution time. Then, we will concentrate on making our tool scale to even bigger datasets by utilizing big data technologies and more experiments with a larger size of datasets. Our tool is an open system and several improvements can be made to increase its efficiency.

## 8. Acknowledgment

This research was financially supported 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, managed by the CMCU (project number 17G1122/ CODE CF 37T03 NJ). The authors would like to thank the OSS experts for their cooperation by providing support, domain knowledge, and environmental data.

## References

- [1] Aarnio, P., Seilonen, I., 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.
- [2] Abbes, H., Gargouri, F., 2018. Mongodb-based modular ontology building for big data integration. *Journal on Data Semantics* 7, 1–27.
- [3] Arp, R., Smith, B., Spear, A.D., 2015. Building ontologies with basic formal ontology. *Mit Press*.
- [4] Athanasiadis, I.N., 2015. Challenges in modelling of environmental semantics, in: International Symposium on Environmental Software Systems, Springer. pp. 19–25.
- [5] Bereta, K., Xiao, G., Koubarakis, M., 2019. Ontop-spatial: Ontop of geospatial databases. *Journal of Web Semantics* 58, 100514.
- [6] Bi, T., Liang, P., Tang, A., 2018. Architecture patterns, quality attributes, and design contexts: How developers design with them, in: 2018 25th Asia-Pacific Software Engineering Conference (APSEC), IEEE. pp. 49–58.
- [7] Bretto, A., 2013. Hypergraph theory. An introduction. *Mathematical Engineering*. Cham: Springer .

- [8] Buttigieg, P.L., Pafilis, E., Lewis, S.E., Schildhauer, M.P., Walls, R.L., Mungall, C.J., 2016. The environment ontology in 2016: bridging domains with increased scope, semantic density, and interoperability. *Journal of biomedical semantics* 7, 57.
- [9] Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G., 2017. Ontop: Answering sparql queries over relational databases. *Semantic Web* 8, 471–487.
- [10] Cyganiak, R., Bizer, C., 2012. D2rq: Accessing relational databases as virtual rdf graphs. URL: <http://d2rq.org/d2r-server>.
- [11] Das, S., 2011. R2rml: Rdb to rdf mapping language. <http://www.w3.org/TR/r2rml/>.
- [12] Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R., 2014. Rml: a generic language for integrated rdf mappings of heterogeneous data, in: *Ldow*.
- [13] Do, H.H., Melnik, S., 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.
- [14] Ehrlinger, L., Wöß, W., 2016. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)* 48, 1–4.
- [15] Ekaputra, F., Sabou, M., Serral Asensio, E., Kiesling, E., Biffl, S., 2017. Ontology-based data integration in multi-disciplinary engineering environments: A review. *Open Journal of Information Systems* 4, 1–26.
- [16] Gibert, K., Horsburgh, J.S., Athanasiadis, I.N., Holmes, G., 2018. Environmental data science. *Environmental Modelling & Software* 106, 4–12.
- [17] Gupta, S., Szekely, P., Knoblock, C.A., Goel, A., Taheriyani, M., Muslea, M., 2012. Karma: A system for mapping structured sources into the semantic web, in: *Extended Semantic Web Conference*, Springer. pp. 430–434.
- [18] Haller, A., Janowicz, K., Cox, S.J., Lefrançois, M., Taylor, K., Le Phuoc, D., Lieberman, J., Garcia-Castro, R., Atkinson, R., Stadler, C., 2018. The sosa/ssn ontology: a joint wec and ogc standard specifying the semantics of sensors observations actuation and sampling, in: *Semantic Web*. IOS Press. volume 1, pp. 1–19.
- [19] Harrison, N.B., Avgeriou, P., 2007. Leveraging architecture patterns to satisfy quality attributes, in: *European conference on software architecture*, Springer. pp. 263–270.
- [20] Kharlamov, E., Mailis, T., Mehdi, G., Neuenstadt, C., Özçep, Ö., Roshchin, M., Solomakhina, N., Soyly, A., Svingos, C., Brandt, S., et al., 2017. Semantic access to streaming and static data at siemens. *Journal of Web Semantics* 44, 54–74.
- [21] Kyzirakos, K., Savva, D., Vlachopoulos, I., Vasileiou, A., Karalis, N., Koubarakis, M., Manegold, S., 2018. Geotriples: Transforming geospatial data into rdf graphs using r2rml and rml mappings. *Journal of Web Semantics* 52, 16–32.
- [22] 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.
- [23] Louge, T., Karray, M.H., Archimède, B., Maamar, Z., Mrissa, M., . Semantic web services composition in the astrophysics domain: Issues and solutions. *Future Generation Computer Systems* 90.
- [24] Masmoudi, M., Karray, M.H., Lamine, S.B.A.B., Zghal, H.B., Archimède, 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.
- [25] Masmoudi, M., Karray, M.H., Lamine, S.B.A.B., Zghal, H.B., Archimède, B., 2020. Memon: Modular environmental monitoring ontology to link heterogeneous earth observed data. *Environmental Modelling & Software* 124, 104581.
- [26] Mountantonakis, M., Tzitzikas, Y., 2019. Large-scale semantic integration of linked data: A survey. *ACM Computing Surveys (CSUR)* 52, 1–40.
- [27] Patroumpas, K., Alexakis, M., Giannopoulos, G., Athanasiou, S., 2014. Triplegeo: an etl tool for transforming geospatial data into rdf triples., in: *Edbt/Icdt Workshops*, pp. 275–278.
- [28] Report, 2019. World economic forum. URL: <https://www.weforum.org/agenda/2019/04/data-oildigital-world-asset-tech-giants-buy-it/>.
- [29] Salmen, D., Malyuta, T., Hansen, A., Cronen, S., Smith, B., 2011. Integration of intelligence data through semantic enhancement .
- [30] 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*, International Society for Optics and Photonics. p. 94640D.
- [31] Vassiliadis, P., Simitis, A., Baikousi, E., 2009. A taxonomy of etl activities, in: *Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP*, pp. 25–32.
- [32] Xiao, G., Ding, L., Cogrel, B., Calvanese, D., 2019. Virtual knowledge graphs: An overview of systems and use cases. *Data Intelligence* 1, 201–223.
- [33] 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](https://doi.org/10.1007/978-3-642-15970-1_6), doi:10.1007/978-3-642-15970-1\_6.