A multi-tiered system for querying the web of data

Ahmed Rabhi, Salah Ouederrou, Rachida Fissoune and Hassan Badir

National School of Applied Sciences, Abdelmalek Essaâdi University, Tangier, Morocco
rabhi.ahmed.1992@gmail.com
salah.ouederrou@gmail.com
ensat.fissoune@gmail.com
hbadir@gmail.com

Abstract. The web of data is a large collection of various data from different domains, these data are distributed on several sources which are heterogeneous and managed independently. Hence, a user may need to query multiple data sources to aggregate pieces of information. In this context, our work aims to set up a system that integrate fragments of information from distributed and independent data sets by providing a single interface to the user. In this paper, we propose a system designed for querying the web of data without having a prior knowledge of the sources contributing to the response. This system adopts a solution for selecting relevant sources in order to decrease network traffic so that the system becomes less dependent on the quality of the connection flow.

1 Introduction

Thanks to Linked Data technologies, the world wide web may enable direct access to raw data, actually Linked Data provides a set of design principles and paradigms for sharing data on the web, thus making the web a global space hosting data in machine-readable format based on standards like RDF and SPARQL. Hence, enabling people and machines to interchange data, which greatly enhance the search for information on the web.

Web of data is a large collection of data from different domains related to each other forming data graphs. According to Heath and Bizer (2011), the web of data is a global space where individuals and organizations have adopted Linked Data standards to publish their data. therefor, the web of data forms a giant graph consisting of billions of RDF data distributed on a large number of Datasets covering all domains such as geography, politics, life science, social networks and others domains. The web of data can be seen as a generic space for sharing containing a variety of data where the user can retrieve information from various domains. These data are stored in RDF datasets and are accessed through sources called Endpoints, the web of data contains nowadays a large number of Datasets like: DBpedia that contain a variety of domains and Bio2RDF that provides a network of linked data for life sciences.

The worry is that the number of data sources has recently increased on a large scale. Moreover, these sources are distributed, heterogeneous and independent one another. Thus, searching information on the web of data means searching information on a number of data sources having no relationship between them. Hence, a user query may require interrogating multiple sources to find a suitable result reporting to his expectation. Thus, the independence of data sources poses a difficulty to answer certain queries which represents a major obstacle for search engines. Therefore, it is necessary to have a system for aggregated search able to collect the appropriate data that responds to a query by integrating fragments of information from several data sources taking into account the distribution and heterogeneity of sources.

In this paper we present a solution to process SPARQL queries over multiple and independent data sources in the web of data. Our work aims to set up a system able to integrate information by aggregating pieces of data from different sources, this solution is based on indexing approach to have a good knowledge about the target data sources which enhances source selection and minimizes communication between data sources and the processing node. This article also highlights the usefulness of Ontology-Based Data Access paradigm to improve query processing by providing domain knowledge and additional vocabulary for query formulation and expose data in a conceptually clear manner.

The rest of this paper is organized as follows: Section 2 presents Basic concepts to be known in this work, Section 3 describes the state-of-the-art in other similar systems and different approaches that might be useful in our work. Section 4 describes our solution and used approaches to process queries. Future directions of our work are presented in Section 5. Finally, Section 7 concludes the article.

2 Basic Concepts and Notation

An RDF statement always consists of three fixed components (Subject, Predicate, Object), the Subject represents a resource, the Predicate can be either a property of the Subject or the representation of a link (a relationship between two resources) and the Object can be either the value of a property or another resource linked to the Subject. Therefore, in RDF, a statement (or triple) represents the smallest unit of data contained in an RDF-type graph, a collection of statements is called an RDF dataset and the web of data contains a large number of these datasets which are accessed via data sources called Endpoints.

Querying RDF data is done using SPARQL (SPARQL Protocol And RDF Query Language) which defines its own syntax for queries, a SPARQL query contains a set of triple patterns, each triple pattern “TP” is a triplet (Subject, Predicate, Object) and each of these elements can be either a resource or a variable. SPARQL provides another type of queries called ASK Queries, the result returned by an ASK Query is boolean, that’s why we use ASK Queries in our work to check the existence or non-existence of a resource.

3 Related work

Several systems were set up to query distributed data sources in the Web Of Data. Quilitz and Leser (2008) present DARQ as a solution that provides a single interface for querying multiple and distributed SPARQL endpoints and makes query federation transparent to the client. FedX is proposed by Schwarte et al. (2011) to allow querying multiple distributed linked data sources as if the data resides in a virtually integrated RDF graph without any local preprocessing using ASK Queries to select sources. Saleem et al. (2013) pay attention in their solution, to the effect of duplicated data on federated querying. The main innovation behind their solution is to avoid querying sources that would lead to duplicated results. The system proposes an index-assisted approach in his process to estimate the overlap between different sources results. Acosta et al. (2011) proposed a solution, called ANAPSID, able to adapt the query execution to the availability of the endpoints and to hide delays from users and reduce execution times. Wang et al. (2013) present LHD as a parallelism-based distributed SPARQL engine. LHD aims to minimize transferred data size through the network as well as to maximize the data transfer rate. The engine uses an optimization algorithm that quickly produces optimal query plans for parallel execution, and a parallel execution system that fully exploits capacity of bandwidth and data sources. Cosmin Basca (2014) presented Avalanche as a solution which is dynamically adaptive to changing external network conditions, provides up-to-date results and is flexible since it makes few limiting assumptions about the structure of participating data sets. Avalanche uses VoID stores to discover relevant sources and collect statistics of the cardinalities (number of instances) for each triple pattern. As avalanche, Görlitz and Staab (2011) and Akar et al. (2012) focus in their works on discovering relevant datasets using voID description that includes information about SPARQL Endpoints and datasets content. The work we present in this paper is an improvement of the solution we proposed in Rabhi et al. (2017).

According to Bagosi et al. (2014), Ontology-Based Data Access (OBDA) is an important approach to access data through a conceptual layer. The main functionality of OBDA systems is query answering, actually, OBDA is a data integration approach that allows users to query several data sources through a unified conceptual view. This paradigm is based on an ontology that plays the intermediary's role between the user and data sources, as stated by Kharlamov et al. (2015). Baader et al. (2016) reported that the general idea is to add an ontology which supplies knowledge of the domain of interest and enriches the necessary vocabulary for queries formulation. Thus, information consumers can have a semantic access to data sets, as cited by Kogalovsky (2012). Calvanese et al. (2017) affirmed that this paradigm allows the user to pose a query by referring only to the ontology without considering data sources that contribute to the response. De Giacomo et al. (2017) affirm that an interesting advantage of OBDA is that it ensures independence between data sources and the ontology, the two levels are only coupled using declarative mappings. This independence is of great importance since it prevents data sources to be modified by users.

4 Our Solution

Searching information must be effective in order to return the maximum possible results, taking into account the large number of data sources, the diversity of datasets and communication between the query node and data sources. In this context, we present a solution to aggregate data from multiple data sources that handles querying a large number of endpoints and the complexity of the user's requests. The idea is to build a search engine designed to query datasets in the Web of Data supporting SPARQL queries and seeks answers over distributed, heterogeneous and independent data sources. The engine aims to minimize network traffic so that the system will be less dependent on network quality and provides a unified view of multiple data sources to the user via a single interface.

As shown in figure 1, our solution is composed of four main Modules: The first one is the Query Analyzer, the system uses this module to prepare the query, its purpose is to analyze different parts of the user's query and to decompose it into sub-queries in order to distribute the

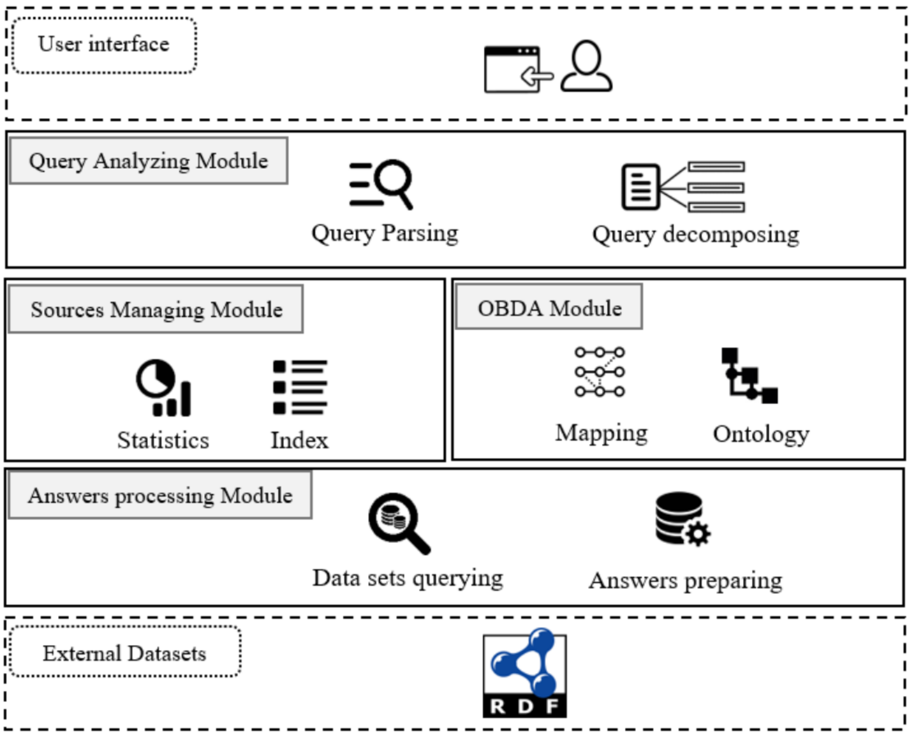

FIG. 1 – *The system architecture*

search of information fragments. Sources Manager Module, uses the index to select relevant sources and stores statistics about sources to rank them depending on their quality. Ontology-Based Data Access module focuses on representing domain knowledge of data sources in a conceptually clear way, unifying several sources in one single view and adapting user queries to be handled by sources. The last module is called Answers Processing, its objective is to evaluate sub-queries in the appropriate data sources, to collect results and executions statistics to finally return results. This section presents the features of each module in more details.

4.1 Query Analyzer

The main functionalities of this module are: parsing the user's query, decomposing it and building sub-queries. Parsing the query is the first step of the process, it consists of reading the SPARQL query which is initially in String format and verifying its syntax, then transforming it into a SPARQL model. Next task is the Query Decomposing, this is an important task in the distributed search, indeed, its aim is to decompose the user's query in order to define the parts of sought information, and identify the different query's parts (triple patterns, variables, filters and aggregate functions). The third task of this module is sub-queries building, the purpose of this task is to create a sub-query for each triple pattern as shown in figure 2, thus, querying data

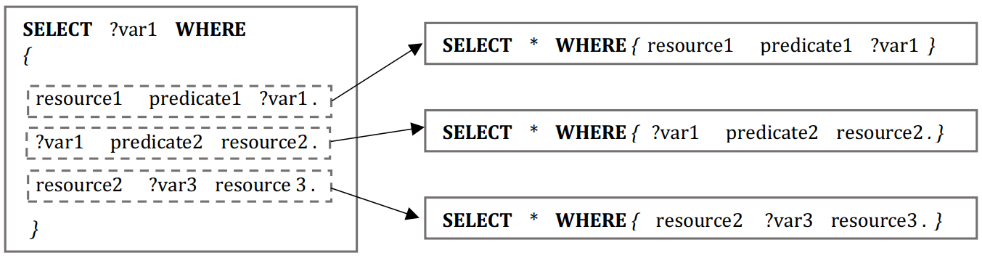
SELECT ?var1 WHERE
{
resource1 predicate1 ?var1 .
?var1 predicate2 resource2 .
resource2 ?var3 resource 3 .
}
SELECT * WHERE { resource1 predicate1 ?var1 }
SELECT * WHERE { ?var1 predicate2 resource2 . }
SELECT * WHERE { resource2 ?var3 resource3 . }

FIG. 2 – *Query Decomposing*

sources returns more results and we'll get more variety of data, which enhance the possibility of responding the user's expectation.

4.2 Sources Manager

The basic purpose of this module is selecting relevant data sources to plan the execution, specifically, this module aims to gather information about the quality of target Endpoints in addition to minimize network traffic between data sources and the processing node. This Module is composed of two main components: the index and statistics data.

The index is used to select relevant data sources contributing to the response in order to access to specific Endpoints and consequently leads to less of non-necessary executions. Actually, an Endpoint may return responses to multiple sub-queries, the problem is that the system would repeat the (connect/disconnect) operations with an Endpoint for each sub-query which make it strongly dependent on the network quality. In our work, we focus on minimizing communication with data sources taking into consideration the couple (Predicate, Object) of triple patterns. Actually, the index stores information about nonexistent resources in data sources in order to avoid querying irrelevant sources and optimizes the execution plan which decreases process time. The main idea of the index is to store (Predicates, Objects) that are unfounded in Endpoints, as shown in figure 3, the index content is a graph structure defining different relationships between Endpoints and resources, the meaning of edges is as follow:

— **noExist**: a relationship between a predicate and an Endpoint, it means that the predicate doesn't exist in the Endpoint.
— **exist**: a relationship between a predicate and an Endpoint, it means that the predicate exists in the Endpoint.
— **noLink**: a relationship between a predicate and an object, it means that the predicate doesn't represent a link between the object and another resource.

So, in this example the Endpoint doesn't contain predicate1 and contains predicate2 but it's not linked to object2.

The index is useful not only for skipping irrelevant data sources but also for sub-queries clustering. In fact, our purpose is to decrease network traffic, so we propose a strategy, based on the index, which consists of grouping sub-queries in order to have a cluster of queries for each target data source. To explain the clustering mechanism, consider, $TP(Q)$ the set of triple patterns of a query Q whereas $TP(Q)$={TP_1, TP_2, $\dots$, TP_n}, TP_i=(s_i p_i, o_i) represents a

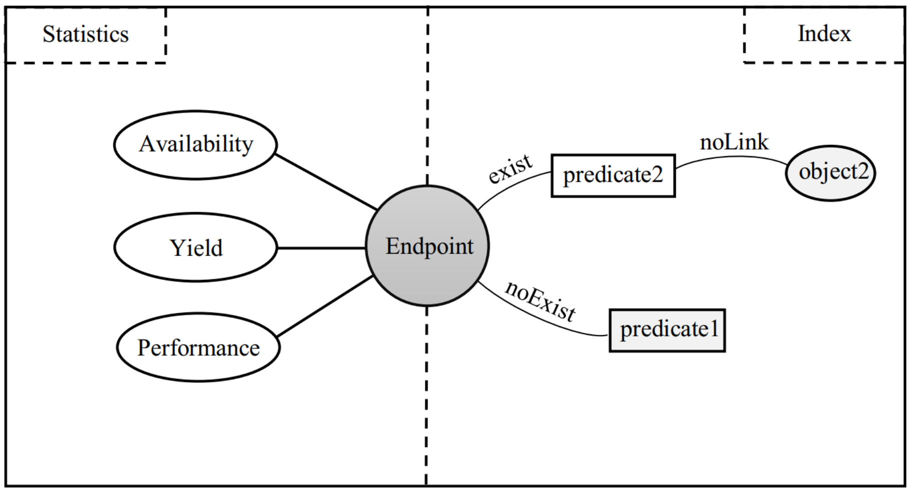

FIG. 3 – *Sources manager*

triplet (Subject, Predicate, Object), and we denote by Q_i the sub-query corresponding to the triple pattern TP_i. After clustering sub-queries **(Algorithm 1)** , we obtain a set C of clusters whereas $C = \{C_1, C_2, ..., C_m\}$ and C_i is the cluster of sub-queries for the source S_i. Besides that, the module manages statistics about data sources (see figure 3), since the improvement of the search engine requires a good knowledge about sources to be queried, their availability, their yield and their efficiency, in addition, the fact of having statistics of the execution history represents a very good benchmark in order to enrich the system over the time.

4.3 Answers processing

Using this module, the system evaluates queries by executing them in the sources according to an execution plan established by the Sources Manager in order to return answers to the user. In addition to processing results, this module interacts with the Sources Manager module to carry out statistics recording and index updates. Actually, preparing final results go trough a set of tasks, as described by **Algorithm 2**, the input is the set of clusters created by **Algorithm 1**, the module executes sub-queries of each cluster and stores received results in a local data storage, then, it performs the necessary updates for the index and the statistics and finally returns the final results to the user. The steps of this process are explained below in more details:

— For each cluster:
— The system establish connection with the data source.
— Executes sub-queries of the clusters.
— If the result is not null, it will be stored in a local dataset.
— If the result is null, the system will verify the existence of the predicate, using an ASK query, and add it in the index if it doesn't exist. And if the predicate exists, the system,

will check the existence of the couple (predicate, object) and add them to the index if the source doesn't contain the couple.

— When all sub-queries of the cluster are executed, the system will be disconnected from the source.
— Updates statistics for the source.
— When all clusters are processed, the system will return finale results.

Algorithm 1 Clustering of sub-queries

```
Input: $S; Q; I$ // Data sources ;Sub-queries ; index
Output: $C$ // Set of clusters
    For each $S_i$ in $S$, do
        $C_i = \{\}$;
        For each $Q_j$ in $Q$, do
            $p$ = predicateOf ($Q_j$);
            $o$ = objectOf ($Q_j$);
            If $(S_i, noExist, p) \in I$, then //$S_i$ doesn't contain $p$
                Continue; // Move to the next sub-query
            Else if $(S_i, exist, p) \in I$, and $(p, noLink, o) \in I$ then
                // $p$ exists in $S_i$ but $p$ is not linked to the resource $o$
                Continue; // Move to the next sub-query
            Else
                $C_i = C_i \cup \{Q_j\}$;
            End if;
        End for ;
        $C = C \cup \{C_i\}$;
    End for;
    Return $C$;
```

Algorithm 2 Results process

```
Input: $C$; //Set of sub-query Clusters
Output: $R_f$; //Final results.
    $R_a = \{\}$; //Result of an execution
    For each $C_i$ in $C$, do
        connectTo($S_i$); // $S_i$ is the source of the cluster $C_i$
        For each $Q_j$ in $C_i$ do // Loop on sub-queries of the cluster
            $p$ = predicateOf ($Q_j$);
            $o$ = objectOf ($Q_j$);
            if(resultsOf($Q_j$) is not NULL), then
                $R_a = R_a \cup$ { resultsOf($Q_j$) };
            Else
                if (ASK($p$) equals False ), then
                    addToIndex($S_i, noExist, p$);
                Else
                    if (ASK($p$,$o$) equals False ), then
                        addToIndex($S_i, exist, p$);
                        addToIndex($p, noLink, o$);
                    End if;
                End if;
            End if;
        End For;
        disconnectFrom($S_i$);
        updateStatistics($S_i$);
    End For;
    $R_f$=Query($R_a$);
    Return $R_f$;
```

5 Work in progress

OBDA module: Since the sources in the web of the data are independent and contain a great variety of data, we adopt the Ontology-Based Data Access paradigm as a solution to enrich datasets querying with an ontology that provides a formal representation of the domains knowledge (Geography, life science, social network) as well as additional vocabulary of different concepts and relationships between them, thus, data will be exposed in a conceptual way. Consequently, the user will be able to query several, independent, distributed and heterogeneous sources by referring only the ontology via a single interface providing a unified view of different data sets. As we presented in our work (Rabhi et al. (2018)), an Ontology-based data access system is composed of three main components: the ontology, the mapping and data layer. The ontology provides a formal semantic description for data sets content, terms of the ontology must be mapped to data layer using mapping assertions to make the correspondence between data sets content and ontology axioms, this mapping associate a query over the conceptual layer (the ontology) to a query over the data layer (external sources), thus, the system will be able to manage queries formulation in order to be compatible for every targeted data source taking into account the query language handled by each source. In addition, the sepa-

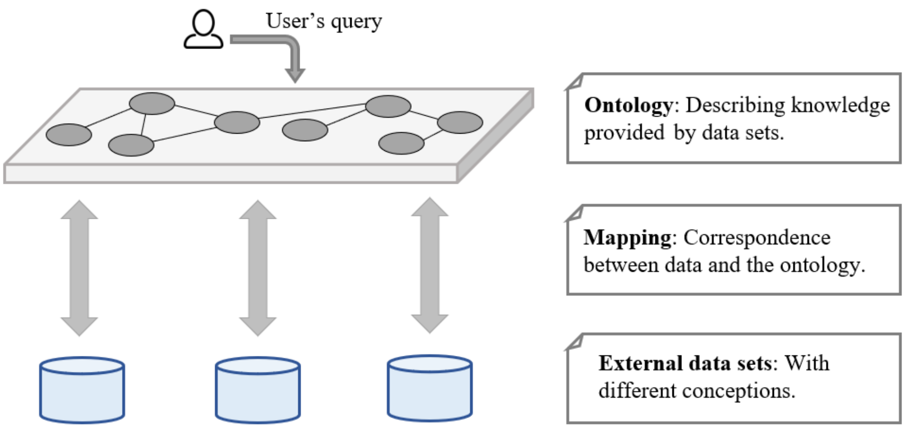
User's query
Ontology: Describing knowledge provided by data sets.
Mapping: Correspondence between data and the ontology.
External data sets: With different conceptions.

FIG. 4 – *OBDA Module*

ration between the data layer and the conceptual layer reflects the independence between the user interface and data sources structure, in this way sources will be prevented from unexpected modifications that may impact another user's searching. The data layer presents data sets to be queried, the physical location of the data, in our work we focus on processing SPARQL queries over the web of data, so, as shown in figure 4, the data layer presents external data repositories reachable on the web via SPARQL Endpoints.

6 Conclusion

In this paper we presented a solution to gather pieces of information from several independent data sources in the Web of Data. In this system, query processing is based on a strategy for sub-queries grouping which aim is to minimize communication between data sources and the query node, using indexing approach. We exposed the paradigm of Ontology-Based data access and its usefulness in providing a unified view of distributed data sets, that are maintained independently, as if they are a single one by presenting a knowledge description of data to mediate between the user's expectations and different data sources structure. Our solution is composed of four main components performing different tasks: query decomposing, selecting data sources and query reformulation and finally answers preparing.

In the future, we hope to complete the development of the Source Manager Module and to implement the OBDA module to improve query processing. Then, we will set up a distributed processing architecture to parallelize the process and distribute it on a cluster of working nodes rather than one computing node, thus, the system will be able to handle the large number of user's requests and the large size of results to be processed. We hope also to make a data sets survey in order to define the domain knowledge of each data set and define axioms of the ontology.

References

Acosta, M., M.-E. Vidal, T. Lampo, J. Castillo, and E. Ruckhaus (2011). Anapsid: an adaptive query processing engine for sparql endpoints. *International Semantic Web Conference*, 18–34.

Akar, Z., T. G. Halaç, E. E. Ekinci, and O. Dikenelli (2012). Querying the web of interlinked datasets using void descriptions. *LDOW 937*.

Baader, F., M. Bienvenu, C. Lutz, and F. Wolter (2016). Query and predicate emptiness in ontology-based data access. *Journal of Artificial Intelligence Research 56 (JAIR) 56*, 1–59.

Bagosi, T., D. Calvanese, J. Hardi, S. Komla-Ebri, D. Lanti, M. Rezk, M. Rodríguez-Muro, M. Slusnys, and G. Xiao (2014). The ontop framework for ontology based data access. *Chinese Semantic Web and Web Science Conference*, 67–77.

Calvanese, D., G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, and G. A. Ruberti (2017). Ontology-based data access and integration.

Cosmin Basca, A. B. (2014). Querying a messy web of data with avalanche. *Journal of Web Semantics 26*, 1–28.

De Giacomo, G., D. Lembo, X. Oriol, D. F. Savo, and E. Teniente (2017). Practical update management in ontology-based data access. *International Semantic Web Conference*, 225–242.

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

Heath, T. and C. Bizer (2011). *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool publishers.

Kharlamov, E., D. Hovland, E. Jiménez-Ruiz, D. Lanti, H. Lie, C. Pinkel, M. Rezk, M. G. Skjæveland, E. Thorstensen, G. Xiao, et al. (2015). Ontology based access to exploration data at statoil. *ISWC*, 93–112.

Kogalovsky, M. R. (2012). Ontology-based data access systems. *Programming and Computer Software 38*(4), 167–182.

Quilitz, B. and U. Leser (2008). Querying distributed rdf data sources with sparql. *European Semantic Web Conference (ESWC)*, 524–538.

Rabhi, A., H. Badir, and A. Rattrout (2018). Towards an ontology-based data access system for aggregated search. *Conference on Advances of Decisional Systems*.

Rabhi, A., S. Ouederrou, and H. Badir (2017). A multi-tiered system for parallel processing of aggregated search over distributed and heterogeneous data sources. *International Conference on Innovation and New Trends in Information Systems*.

Saleem, M., A.-C. N. Ngomo, J. X. Parreira, H. F. Deus, and M. Hauswirth (2013). Daw: Duplicate-aware federated query processing over the web of data. *ISWC*, 574–590.

Schwarte, A., P. Haase, K. Hose, R. Schenkel, and M. Schmidt (2011). Fedx: Optimization techniques for federated query processing on linked data. *International Semantic Web Conference*, 601–616.

Wang, X., T. Tiropanis, and H. C. Davis (2013). Lhd: Optimising linked data query processing using parallelisation.

Résumé

Le web des données est une grande collection de données variés et de domaines différents, ces données sont distribuées sur des sources hétérogènes et gérées indépendamment. Par conséquent, un utilisateur peut avoir besoin d'interroger plusieurs sources de données pour agréger les pieces de l'informations. Dans ce contexte, notre travail vise à mettre en place un système qui intègre des fragments d'informations provenant de data sets distribués et indépendants en fournissant une interface unique à l'utilisateur. Dans cet article, nous proposons un système conçu pour interroger le web des données sans avoir une connaissance préalable sur les sources contribuants à la réponse. Ce système adopte une solution pour sélectionner les sources pertinentes afin de diminuer le trafic réseau de sorte que le système devienne moins dépendant de la qualité du réseau.