ELITE: An *E*ntai*l*ment-based Federated Query Engine for Complete and Transparent Semantic Data *Inte*gration

Andreas Nolle and German Nemirovski

Albstadt-Sigmaringen University, Jakobstraße 6, 72458 Albstadt, Germany
{nolle,nemirovskij}@hs-albsig.de

Abstract. In recent years the core of the semantic web has evolved into a conceptual layer built by a set of ontologies mapped onto data distributed in numerous data sources, interlinked, interpreted and processed in terms of semantics. One of the central issues in this context became the federated querying of such linked data. This paper presents the federated query engine ELITE that facilitates a complete and transparent integration and querying of distributed autonomous data sources. To achieve this aim a combination of existing approaches for Ontology-based Data Access (OBDA) and federated query processing on Linked Open Data (LOD) is applied. Consolidating technologies like entailment regimes, the DL-Lite formalism, query rewriting, mapping relational data to RDF and an improved implementation of R-Tree based indexing contributes to the unique features of this federation engine. ELITE thereby enables the integration of various kinds of data sources, for example as relational databases or triple stores, simplicity of query design, guaranteed completeness of query results and highly efficient query processing. The federation engine has been developed and evaluated in the domain of carbon reduction in urban planning.

Keywords: Federated Querying, Entailment Regime, Query Rewriting, Ontology-based Data Access (OBDA), DL-Lite, OWL 2 QL, SPARQL, Linked Open Data (LOD), Indexing, R-Tree Index

1 Introduction

Due to initiatives like Linking Open Data (LOD) [1] the total amount of semantic data publicly available online grows continuously. In recent time the querying of such distributed data has become an issue attracting significant attention within semantic (web) research. The Approaches emerging in the context of semantic data integration range from systems for processing federated queries on linked RDF data to ontology-based data access (OBDA) focusing on linking relational data to ontologies.

Systems and methodologies for federation of linked data, that are basically stored in RDF triple stores exposing a SPARQL service, belong to the first category. Federation is one of three main paradigms for the design of a linked data infrastructure identified by Görlitz & Staab [2]. Its aim is to analyze and process the query initially formulated by the client in order to i) identify query parts related to single sources, ii) rewrite the original query for each source, iii) forward the rewritten queries to the corresponding sources and iv) merge the query results carried out by each source. The main obstacle of this approach is the requirement to operate with a set of vocabularies referred by the distributed sources like DBPedia [3], FOAF [4] or YAGO [5]. Consequently, most systems implementing the federation approach expect that clients formulate queries by using all vocabularies related to each integrated data source. This fact restricts the flexibility of clients and forces users to think in terms that are alien to their usual domain of discourse.

The approaches of the second category aim at building a service tending to present a conceptual view of the domain of discourse formulated in terms of an ontology that is univocally mapped onto the data layer. Clients accessing the service don't have to know how the data layer is organized, formatted, and structured. They formulate semantic queries referring to the ontology and hence, to the native vocabulary of the domain of discourse. The conceptual view typically consists of the ontology TBox only. There is no need for storing ABox individuals since semantic queries are translated in runtime into the query language (usually SQL) and are comprehensible for the data sources comprising the data layer. Furthermore, application of formalisms designed especially for OBDA purposes facilitates effective query rewriting e.g. from SPARQL into SQL, and therefore high efficiency of query evaluation ensuring sound and complete query results. However, since existing OBDA approaches focus on accessing (mostly non-RDF) data in relation to its semantics, they don't take into account sources that already provide RDF data via SPARQL interfaces like triple stores. [6–8]

The approach proposed in this paper consolidates the advantageous features of the two approaches described above and is free of their most significant disadvantages. It uses the conceptual view for the domain of discourse formally specified as ontology, in a way similar to the OBDA approach. The central ontology in turn is linked and mapped onto vocabularies of each single data source integrated into the federated infrastructure. However in contrast to OBDA systems our approach exploits query execution strategies described in federation approaches for linked data combined with reasoning applied centrally to each query launched by the client. Due to the described architecture the system guarantees completeness of query results and enables clients to operate on the vocabulary strongly related to the domain of discourse. Besides, the system supports highly efficient query processing by means of an improved implementation of R-Tree-based indexing. The proposed approach has been implemented and evaluated for the domain of carbon reduction in urban planning.

The document is structured as follows: After Section 2 containing a survey of related work we introduce the system architecture of our approach in Section 3. The implementation of entailment regimes through rewriting of SPARQL queries is described in Section 4. Subsequently, Section 5 presents an improved R-Tree-based indexing. Before concluding this paper in Section 7 we discuss evaluation results in Section 6.

2 Related Work

This paper is written for readers familiar with the paradigm of semantic web [9], basics of knowledge representation by means of first order logics and description logics (DLs) [10] and, having experience in application of SPARQL [11]. We recommend the mentioned references for more information about these topics.

In order to achieve complete results on SPARQL queries not only the explicitly specified data and its relations have to be taken into account but also the knowledge that can be inferred by reasoning on the RDF graph. The definition of such extended interpretation on query evaluation is part of the proposed recommendation of SPARQL 1.1 and is called entailment regimes. [12,13]

As stated in the introduction, ODBA is one of the approaches fundamental to the work described in this paper. It enables a mapping of relational data to ontologies and therefore access to this data by reference to its semantics. The OBDA platforms –ontop– [14] and MASTRO-I [15,16] facilitate highly efficient query evaluation through a syntactically restricted ontology language. Through this they enable reasoning for complete query results with reduced complexity as well as highly optimized techniques for query rewriting. In contrast, other RDB2RDF tools merely focusing on the real-time conversion of relational data to RDF, commonly have restricted support of entailment regimes and weak performance in query answering especially for large data sets. Apart from the most popular D2RQ platform [17,18] tools like Virtuoso RDF views [19], Triplify [20] or Revelytics Spyder [21] also belong to these systems. Extended but not complete lists of other systems can be found online. [14,22–24]

Another approach strongly related to the present work focuses on facilitating the infrastructure for linked data, and in particular on optimization of federated SPARQL queries. One of the most complete surveys of this issue is given by Görlitz & Staab [2]. The authors distinguish between different architecture variations for implementing a query-based search on LOD. As the other architectures like peer-to-peer systems have crucial disadvantages in connection with LOD and the task to query its sources, we will only consider the federation architecture. Due to the lack of space in this paper we don't focus on other architectures mentioned by Görlitz & Staab. In federative systems only metadata, indices and/or data statistics of each data source are kept centrally and represent data stored in each source. These items are processed to guarantee completeness of query results on the one hand, and to facilitate high efficiency of querying distributed linked (open) data sources on the other hand. Currently available are several implementations of this approach, such as FedX [25,26], SemWIQ [27], DARQ [28], SQUIN [29], UniStore [30], SPLENDID [31], ANAPSID [32], Avalanche [33] or AliBaba [34]. Apart from these systems, federated queries can be processed by endpoints implementing the SPARQL version 1.1. SPARQL 1.1 has been extended by a construct for specification of federated queries [35]. However most of the systems mentioned above as well as SPARQL 1.1 engines accept only federated queries, referring to the same vocabularies which are referred in the data sources to be queried. This fact puts strong limitations on usability for clients that formulate queries and therefore have to handle all these vocabularies.

To our best knowledge at least six approaches exist that adopt various techniques to overcome this limitation. For instance, the approach described by Vidal et al. [36] uses explicit rule definitions to map elements of the domain specific ontology to the central one. However due to the formalism used for the rule specifications the application of entailment regimes is not considered in its entirety. Another approach by Correndo et al. [37] uses RDF to express rewriting rules and enables query rewriting for each single source. This approach does not require a central ontology. Instead it enables translations among ontologies referred in the sources to be queried. The third system called LOQUS [38] provides a conceptual view (central ontology) to the distributed sources. LOQUS rewrites the original query into a set of source specific ones by using links and mappings specified in the central ontology, but doesn't take into account entailment regimes. The advancement of this system called ALOQUS introduced by Joshi et al. [39] facilitates the querying of linked data based on ontology alignments without the need to have detailed information about the data sources. ALOQUS additionally supports an automatic mapping between ontologies to get the alignments but also omits reasoning steps. A further approach elaborated by Li & Heflin [40,41] uses a reasoner to produce results related to a query and therefore ensure completeness. However since the reasoner is located centrally and is invoked at the end of the query execution process all of the data selected has to be entirely loaded in the reasoner. The sixth approach contributed by Makris et al. [42,43] utilizes ontology mappings to rewrite original queries in terms of the target ontologies in order to access federated sources and is therefore the closest one to our approach.

Despite our extensive studies and to the best of our knowledge none of the existing approaches considers the complexity of reasoning and query answering. Concerning this Calvanese et al. [7,8] propose the description logics family DL-Lite with the aim of low reasoning complexity. They have shown that reasoning tasks for DL-Lite are in PTIME and query answering is in LOGSPACE (more precisely even in AC^0), each in size of the TBox and ABox, respectively. DL-Lite facilitates these efficient reasoning and query answering features over large data sets with maximum applicable expressiveness. In contrast, reasoning on more expressive DLs is in worst-case EXPTIME-hard and query answering co-NP-hard in the size of the ABox (data complexity). Since the OWL 2 QL profile [44] bases upon the DL-Lite family, we use OWL 2 QL as specification language of the central ontology. Through that we achieve not just an efficient rewriting of the original query into data source specific queries but also an efficient accessing of relational data sources using OBDA tools like –ontop– and its integrated reasoner Quest [14,45,46]. Furthermore we don't require any additional definition of mapping or transformation rules like other mentioned approaches.

3 System Architecture

The proposed architecture has been developed with the goals mentioned in the introduction (Section 1). These are i) efficient query processing in particular complete evaluation of conjunctive federated queries, ii) ability for clients to formulate queries abstracted from the technological and structural characteristics of the data sources and iii) ability to federate data sources using different schemas, e.g. RDF triple stores and relational data base systems. One key feature of the proposed federative approach is the application of a single central ontology that on the one hand represents the vocabulary and knowledge structure of the domain of discourse and, on the other hand serves as a mediator for heterogeneous data sources. All queries formulated by a client and potentially targeting data distributed in single sources refer to this ontology. However the client is not constrained to use only the central vocabulary but is also able to use elements (e.g. concepts) of data source specific vocabularies. The system architecture of ELITE is outlined in Fig. 1.

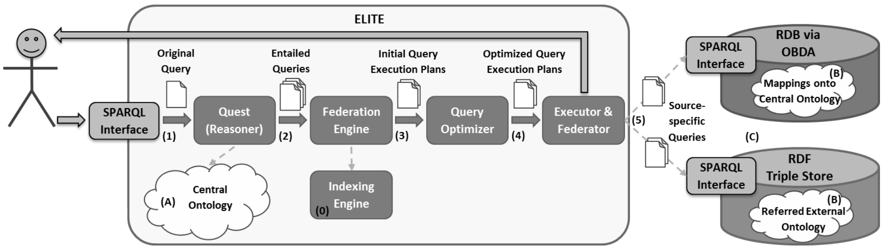


Fig. 1. System Architecture of ELITE

The central ontology (A in Fig. 1) is basically a TBox that comprises apart from own items, i.e. concepts, roles and axioms defined in its own namespaces, items semantically related to the domain of discourse and specified in openly accessible vocabularies (B), e.g. those used in the LOD sources. The integration of such items into the central ontology is implemented by their linking through using RDFs or OWL properties. For instance by equating the concept `ResidentialBuilding` of the central ontology with the concept `DwellingHouse` of an external ontology (`ResidentialBuilding` ≡ `DwellingHouse`), or by subsumption of an external concept `DwellingHouse` (`DwellingHouse` ⊑ `Building`).

Such mapping and linking facilitate federated querying of related data sources using their own vocabularies. Issues related to the design of the central ontology are out of scope of this paper, we just would like to mention that the efficient design of the central ontology is basically carried out manually, e.g. using a dedicated process model, as it is described by Nemirovski et al. [47], or using automated ontology matching systems as shown by Shvaiko & Euzenat [48].

The proposed approach works for triple stores as well as for relational databases under the condition that all integrated sources (C) provide a SPARQL-endpoint to query data. For conversion of relational data to the RDF format RDB2RDF tools like D2RQ and OBDA tools like –ontop– are used. These tools are applied locally for each source to map relational data to items of the central ontology as it is shown for example by Poggi et al. [6]. Apart from the central ontology (A) another centrally located component of ELITE architecture is the index that is initialized (0) after the centralized ontology and the data sources are defined. The most important principles of index structure and its usage for identification of data sources related to a query are described in Section 5.

The querying process starts when a query formulated by an external application or by a user is launched at the central SPARQL endpoint of ELITE (1). Elements of this query, e.g. basic graph patterns (BGP), refer to items of the central ontology and elements of other vocabularies linked to the central ontology as described above. In the first step the initial query is rewritten (2) by means of entailment regimes, whereby the semantics of the central ontology as well as of the linked external ontologies are taken into account. The application of entailment regimes (described in Section 4) results in multiple queries reflecting all possible ways of expressing the same statement using the knowledge implicated by all considered vocabularies. The disjunction of evaluation results of these queries by each data source delivers the initial query result complete in terms of all considered vocabularies. Yet not all BGPs contained in these queries are relevant to all integrated sources. Therefore not all queries, more precisely its patterns generated through application of entailment regimes, can be answered by any sources. To avoid sending queries causing empty results, such queries are excluded from further processing by the index look-up procedure. In the next step (3) this procedure is applied for every single BGP in all queries to determine the sources which contain data related to each BGP. The index also delivers an estimated size of results that would be returned by each data source for each BGP (see Section 5 for a detailed description of indexing). Using estimated sizes and dependencies among patterns of each query, an optimized query execution plan is generated (4). After results to all pre-selected queries executed by pre-selected sources according to the previously generated plan are retrieved, the federated result is returned to the client that has launched the initial query (5).

4 Entailment by Query Rewriting

In order to take knowledge inferred from persistently stored RDF triples in the query execution process into account the implementation of such entailment regimes can be done in three different ways. The first technique is to materialize all triples that are inferred according to the used entailment regimes e.g. RDFS entailment regimes, that is also called forward-chaining. The alternative technology is called backward-chaining. It operates by inferring of new triples through query rewriting in runtime. The third option is a hybrid combination of both approaches. These techniques (query rewriting and materialization of inferred triples) are further described by Glimm [13]. Though query evaluation using forward-chaining can be more efficient because e.g. no further reasoning is required at all [49], it isn't really compatible with the federation approach. Since most of linked (open) data sources grant read only access to the clients, an external location for storing the inferred triples has to be found. Furthermore all changes of the data sources have to be reflected by the materialized data to keep it up to date. Contrary to this backward-chaining is more flexible with respect to handling dynamic data and without the need of an additional location to store the materialized data.

Since in our approach the entailment is processed centrally, all inferable knowledge as well as all conceivable external concepts and properties referenced in the central ontology are taken into account in the patterns of the rewritten SPARQL queries. Therefore the federative system doesn't have to rely on the entailment of single data sources to guarantee the completeness of query results. Even if local SPARQL services of e.g. some RDB2RDF tools don't implement entailment regimes at all, the query results will be complete in terms of the central ontology. Yet the central query rewriting does not only help in cases when local SPARQL services are not (fully) support entailment. Furthermore, even though a SPARQL service of a single source implements the entailment regimes, it only can infer knowledge based on the vocabularies (ontologies) available at the corresponding source. If the central ontology that specifies the domain of discourse is not among these vocabularies, completeness of query results in terms of the central ontology can't be guaranteed based on local entailments only.

At this point we want to thank the developers of Quest who provided us with the source code of the system at our disposal. We extended it by implementing a translator to convert datalog programs, Quests internal rule representation of the rewritten queries into SPARQL queries by analyzing and inspecting the generated rules. As Quest contains various rewriting implementations we have chosen the Tree Witness Rewriter implemented by Kontchakov et al. [50–52] because the authors have shown that this rewriter produces better evaluation results than to other ones. Assuming for example that the TBox of the central ontology contains the following statements (namespace specifications are omitted):

$$\begin{array}{l}
\texttt{ResidentialBuilding} \sqsubseteq \texttt{Building} \\
\texttt{CommercialBuilding} \sqsubseteq \texttt{Building} \\
\exists \texttt{hasBuilding\_Floor\_Area} \sqsubseteq \texttt{Building} \\
\exists \texttt{hasBuilding\_Floor\_Area}^{-} \sqsubseteq \texttt{Building\_Floor\_Area} \\
\exists \texttt{hasValue} \sqsubseteq \texttt{Building\_Floor\_Area} \\
\texttt{Range(hasValue)} \equiv \texttt{rdf:decimal} \\
\exists \texttt{hasResBuildingFloorArea} \sqsubseteq \texttt{ResidentialBuilding} \\
\exists \texttt{hasResBuildingFloorArea}^{-} \sqsubseteq \texttt{ResidentialFloorArea} \\
\texttt{ResidentialFloorArea} \sqsubseteq \texttt{Building\_Floor\_Area} \\
\texttt{hasResBuildingFloorArea} \sqsubseteq \texttt{hasBuilding\_Floor\_Area}
\end{array} \tag{1}$$

The distributed sources $\mathrm{DS}_A(2)$ and $\mathrm{DS}_B(3)$ contain the following data (ABox):

```
ResidentialBuilding(Hundertwasserhaus)                                (2)
ResidentialFloorArea(Hundertwasserhaus3356)
:Hundertwasserhaus :hasResBuildingFloorArea :Hundertwasserhaus3356
:Hundertwasserhaus3356 :hasValue 3356.0^^xsd:decimal
:Casa_Mila :hasResBuildingFloorArea :Casa_Mila1000
:Casa_Mila1000 :hasValue 1000^^xsd:decimal
```

```
CommercialBuilding(Tanzende_Tuerme)                                   (3)
Building_Floor_Area(Tanzende_Tuerme33357)
:Tanzende_Tuerme :hasBuilding_Floor_Area :Tanzende_Tuerme33357
:Tanzende_Tuerme33357 :hasValue 33357.0^^xsd:decimal
```

To get all buildings the user only has to define the following query:

```
SELECT ?building { ?building a :Building . }                          (4)
```

After entailment by query rewriting we get the following datalog program:

```
q(building) :- hasResBuildingFloorArea(building,_)                    (5)
q(building) :- Building(building)
q(building) :- CommercialBuilding(building)
q(building) :- hasBuilding_Floor_Area(building,_)
q(building) :- ResidentialBuilding(building)
```

These programs can be rewritten accordingly as SPARQL queries that have to be federated. To get all floor area values of all buildings the user only has to define the following query:

```
SELECT ?building ?buildingFloorArea {                                 (6)
  ?building :hasBuilding_Floor_Area _:bfa .
  _:bfa :hasValue ?buildingFloorArea . }
```

It is important to notice that in a case when entailment regimes are not implemented at SPARQL services, neither centrally nor locally, it is the task of the client to formulate the query in a way that makes sure that all related RDF graph patterns are taken into account. For this purpose the client has to be aware of all formal vocabulary structures her/his query refers to, and the semantics that can be applied to these vocabularies. Consequentially the complexity of queries and of client applications would increase significantly. For instance, the query for selecting all floor area values of all buildings (6) would look like this:

```
SELECT ?building ?buildingFloorArea {                                 (7)
    { ?building :hasBuilding_Floor_Area _:bfa . }
  UNION
    { ?building :hasResBuildingFloorArea _:bfa . }
  _:bfa :hasValue ?buildingFloorArea . }
```

5 Indexing

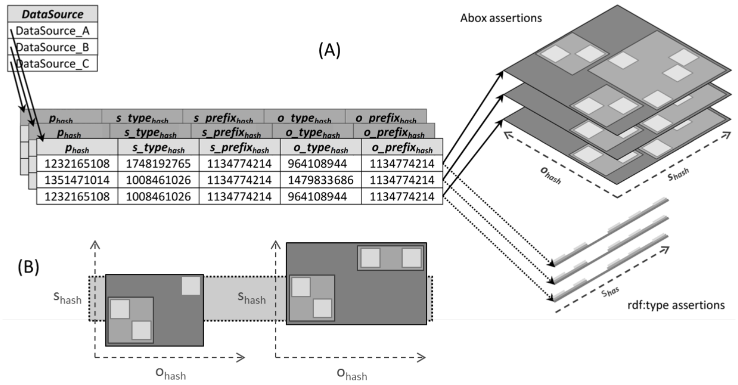
Since a query may address complex relations across several data sources it is not unusual that a single source is only able to answer a part of the query. However the complete query can't be forwarded to a source that is not able to deliver results for each query pattern and hence would return an empty result set. Instead, the federation engine has to identify relations between data sources and query patterns, to generate, based on these relations, dedicated sub-queries and forward them to corresponding sources. To select all sources relevant to each single part of a query an index catalog implementing a look-up service and able to estimate the size of the sub-query results of each source is required.

A comparative analysis of currently available indexing methods for distributed RDF data can't be provided in this paper due to space limitations. For a survey of the state of the art we point out the work of Görlitz & Staab [2]. Here we only describe the indexing approach developed for ELITE. Our index is an extension of the approach called QTree developed by Harth et al. [53]. QTree provides an optimized index to summarize RDF data. The index is based on R-Trees [54] and summarizes the data in so called Minimum Bounding Boxes (MBBs). MBBs are approximation cubes of three-dimensional points built with the hash values for each triple element (subject, predicate and object). The query result estimation for single SPARQL operations like joins is done by performing the same operation on the MBBs. Afterwards, the results are considered for selecting data sources related to particular BGPs of a query and for determining the sequence of sub-query processing (query execution plans). Since the approach of QTree has some limitations, e.g. inaccurate granularity of MBBs, and therefore is only able to process simple queries for small sets of data this approach has been adapted by Prasser et al. [55] who have implemented PARTree, a more efficient indexing suited for complex querying of large RDF data sources.

Similar to PARTree our approach follows the strategy to index each single data source independently but in contrast to PARTree it uses more fine-grained partitioning of RDF triples. A systematical representation of the described index architecture is shown below in part (A) of Fig. 2. The index partitions are identified by the hash value of the predicate's URI (p_{hash}), the hash value of the prefix - more precisely the namespace - of the subject (s_prefix_{hash}) and object (o_prefix_{hash}) and additionally by the type of the subject (s_type_{hash}) and object (o_type_{hash}). In this context the type of a particular ABox individual is defined as the URI of the TBox concept connected to the individual by the `rdf:type` relation. For the last triple of the assertion below the type of the subject `Hundertwasserhaus` would be `ResidentialBuilding`:

```
ResidentialBuilding(Hundertwasserhaus)                                    (8)
FloorArea(Hundertwasserhaus3356)
:Hundertwasserhaus :hasResBuildingFloorArea :Hundertwasserhaus3356
```

If no type of an individual can be derived the assigned hash value for s_type_{hash} or o_type_{hash} is 0. Since the `rdf:type` for TBox elements does not exist, this would be also the case for the objects in type assertions like the first two statements of the example above. Using this fine-grained segmentation for triple partitions leads to more accurate results compared to QTree as well as PARTree and therefore to a more exact data source selection as well as more precise result estimations. For type assertions like e.g. the first one shown in (8) the described segmentation results in an one-dimensional index structure, because the type URIs of the predicate and the object as well as the namespaces for the subject are unique in each partition.



DataSource
DataSource_A
DataSource_B
DataSource_C

p_{hash}	s_type_{hash}	s_prefix_{hash}	o_type_{hash}	o_prefix_{hash}
1232165108	1748192765	1134774214	964108944	1134774214
1351471014	1008461026	1134774214	1479833686	1134774214
1232165108	1008461026	1134774214	964108944	1134774214

Fig. 2. Systematical Representation of our Index Structure & Spatial Join of two BGPs

Calling the index look-up service for a BGP results in selections of MBBs for each source containing triples fitting to this pattern. For this purpose the corresponding hash values are calculated for all pattern elements defined by an URI and all found MBBs of each source are returned. For instance two BGBs that are connected with the same subject variable, estimated results can be calculated by joining the spatial results (MBBs). A graphical representation of this example is given in part (B) of Fig. 2.

Besides the partitioning another main difference to PARTree and QTree is that our index isn't held in main memory but instead stored in a spatial database. Due to this solution we benefit from the optimized features of spatial database systems to manage spatial objects, to perform spatial operations like joins and to achieve high scalability of the index. To fill the index with summarized data of each data source we crawl them through to SPARQL queries specially constructed for this purpose.

6 Evaluation

The goal of our evaluation was to prove the completeness of query results generated by ELITE, and furthermore to demonstrate that conventional systems, which neither use query rewriting techniques nor apply materialization of inferred data aren't able to deliver comparable results. To evaluate ELITE we have used part of an ontology developed for the SEMANCO project [56] operating in the domain of carbon reduction in urban planning. This ontology conforms to the rules defined by the DL-Lite$_{\mathcal{A}}$ formalism. DL-Lite$_{\mathcal{A}}$, introduced by Poggi et al. [6], is a member of the DL-Lite family and comprises the computational feature of the DL-lite core variant with higher expressiveness.

Since none of the six strongly related approaches mentioned in Section 2 are publicly accessible we have compared ELITE to available state of the art federation systems with respect to completeness of query results. For evaluation purposes several data sets containing data properties of individuals specified within the selected ontology have been distributed over two data sources. Utilizing D2RQ we mapped the relational data to the appropriated ontology elements.

Using this infrastructure a comparative evaluation with the FedX system for federation of distributed data has been carried out. As stated in Section 4, query results generated by D2RQ using FedX only take into account data defined explicitly by the D2RQ mappings. This hypothesis has been confirmed by the evaluation results. Since inference has been carried out neither by local installations of D2RQ nor by FedX, no other matches to the query patterns originally submitted have been identified. In contrast to this, caused by the implemented entailment by query rewriting, ELITE has delivered complete results taking into account all implicit knowledge. As expected, evaluation results (Table 1) demonstrate that ELITE provides an effective method for entailment-based federated query processing and delivers complete results when other systems fail.

Table 1. Selected Evaluation Results

	FedX	**ELITE**	**subsistent**
query 1 (coding (4))	20	40	40
query 2 (coding (6))	10	20	20
query 3 (coding (7))	20	20	20

7 Conclusion and Future Work

This paper introduced the federated query engine ELITE that enables querying of linked data distributed over a set of autonomous data sources. Application of entailment regimes by query rewriting facilitates completeness of query results in terms of a central ontology that represents a native vocabulary of the domain of discourse. By the ability to generate complete query results regardless of the SPARQL endpoint features supported by integrated data sources, ELITE has an outstanding position compared to other federation approaches. Beside that this paper elucidated an improved implementation of R-Tree based semantic indexing. This technology and application of the DL-Lite formalism for ontology design are crucial in relation to high efficiency.

Our next steps towards further improvement of ELITE characteristics will be the implementation of a query execution plan, optimization and identification of alternative techniques for federated querying. A further purpose is the proof of soundness of query results that are delivered by the rewritten queries. Since in this paper we only have described the evaluation results concerning completeness of query results, our future target will be the evaluation of efficiency. We will evaluate selected benchmarks and try to quantify the contributions of particular technologies to the efficiency of query processing.

Acknowledgments. We thank Dr. Mariano Rodríguez-Muro for providing us the source code of Quest as well as for supporting. The main contribution of this work has been developed within the SEMANCO project, which is being carried out with the support of the Seventh Framework Programme "ICT for Energy Systems" 2011–2014, under the grant agreement no. 287534.

References

1. LinkingOpenData - W3C Wiki. http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
2. Görlitz, O., Staab, S.: Federated data management and query optimization for linked open data. In: New Directions in Web Data Management 1. Springer (2011) 109–137
3. DBpedia. http://dbpedia.org
4. The Friend of a Friend (FOAF) project. http://www.foaf-project.org
5. YAGO2s: A High-Quality Knowledge Base. http://mpii.de/yago
6. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. In: Journal on data semantics X. Springer (2008) 133–173
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated reasoning **39**(3) (2007) 385–429
8. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. Journal of Artificial Intelligence Research **36**(1) (2009) 1–69
9. Hitzler, P., Krotzsch, M., Rudolph, S.: Foundations of semantic web technologies. Chapman and Hall/CRC (2011)
10. Baader, F.: The description logic handbook: theory, implementation, and applications. Cambridge: Cambridge University Press (2003)
11. SPARQL 1.1 Overview. http://www.w3.org/TR/sparql11-overview
12. SPARQL 1.1 Entailment Regimes. http://www.w3.org/TR/sparql11-entailment
13. Glimm, B.: Using SPARQL with RDFS and OWL entailment. In: Reasoning Web. Semantic Technologies for the Web of Data. Springer (2011) 137–201
14. Rodrıguez-Muro, M., Calvanese, D.: High performance query answering over DL-Lite ontologies. In: Proc. of the 13th Int. Conf. KR. (2012)
15. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R.: MASTRO-I: Efficient integration of relational data through DL ontologies. In: Proceedings of the 20th International Workshop on Description Logics, Citeseer (2007)
16. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The Mastro system for ontology-based data access. Semantic Web **2**(1) (2011) 43–53
17. Bizer, C., Seaborne, A.: D2RQ-treating non-RDF databases as virtual RDF graphs. In: Proceedings of the 3rd international semantic web conference (ISWC2004). (2004) 26
18. Bizer, C., Cyganiak, R.: D2r server-publishing relational databases on the semantic web. In: 5th international Semantic Web conference. (2006) 26
19. Mapping Relational Data to RDF with Virtuoso's RDF Views. http://virtuoso.openlinksw.com/whitepapers/relational%20rdf%20views%20mapping.html

20. Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumueller, D.: Triplify: light-weight linked data publication from relational databases. In: Proceedings of the 18th international conference on World wide web, ACM (2009) 621–630
21. Spyder. `http://www.revelytix.com/content/spyder`
22. Implementations - RDB2RDF. `http://www.w3.org/2001/sw/rdb2rdf/wiki/Implementations`
23. Links and Resources. `http://d2rq.org/resources`
24. Rodrıguez-Muro, M., Calvanese, D.: Quest, a system for ontology based data access. In: OWLED-2012 - OWL: Experiences and Directions Workshop 2012. Volume 849., CEUR Electronic Workshop Proceedings (2012)
25. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: a federation layer for distributed query processing on linked open data. In: The Semanic Web: Research and Applications. Springer (2011) 481–486
26. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: Optimization techniques for federated query processing on linked data. In: The Semantic Web–ISWC 2011. Springer (2011) 601–616
27. Langegger, A., Wöß, W., Blöchl, M.: A semantic web middleware for virtual data integration on the web. In: The Semantic Web: Research and Applications. Springer (2008) 493–507
28. Quilitz, B., Leser, U.: Querying distributed RDF data sources with SPARQL. In: The Semantic Web: Research and Applications. Springer (2008) 524–538
29. Hartig, O., Bizer, C., Freytag, J.C.: Executing SPARQL queries over the web of linked data. In: The Semantic Web-ISWC 2009. Springer (2009) 293–309
30. Karnstedt, M., Sattler, K.U., Hauswirth, M.: Scalable distributed indexing and query processing over Linked Data. Web Semantics: Science, Services and Agents on the World Wide Web **10** (2012) 3–32
31. Görlitz, O., Staab, S.: SPLENDID: SPARQL endpoint federation exploiting VOID descriptions. In: Proceedings of the 2nd International Workshop on Consuming Linked Data, Bonn, Germany. (2011)
32. Acosta, M., Vidal, M.E., Lampo, T., Castillo, J., Ruckhaus, E.: ANAPSID: AN Adaptive query ProcesSing engIne for sparql enDpoints. In: The Semantic Web–ISWC 2011. Springer (2011) 18–34
33. Basca, C., Bernstein, A.: Avalanche: putting the spirit of the web back into semantic web querying. In: The 6th International Workshop on SSWS at ISWC. (2010)
34. AliBaba. `http://www.openrdf.org/alibaba.jsp`
35. SPARQL 1.1 Federated Query. `http://www.w3.org/TR/sparql11-federated-query`
36. Vidal, V.M., de Macêdo, J.A., Pinheiro, J.C., Casanova, M.A., Porto, F.: Query processing in a mediator based framework for linked data integration. International Journal of Business Data Communications and Networking (IJBDCN) **7**(2) (2011) 29–47
37. Correndo, G., Salvadores, M., Millard, I., Glaser, H., Shadbolt, N.: SPARQL query rewriting for implementing data integration over linked data. In: Proceedings of the 2010 EDBT/ICDT Workshops, ACM (2010) 4
38. Jain, P., Verma, K., Yeh, P.Z., Hitzler, P., Sheth, A.P.: LOQUS: Linked Open Data SPARQL Querying System. Technical report, Tech. rep., Kno. e. sis Center, Wright State University, Dayton, Ohio, 2010. Available from http://www. pascal-hitzler. de/resources/publications/loqus-tr-2010. pdf (2010)

39. Joshi, A.K., Jain, P., Hitzler, P., Yeh, P.Z., Verma, K., Sheth, A.P., Damova, M.: Alignment-based Querying of Linked Open Data. In: On the Move to Meaningful Internet Systems: OTM 2012. Springer (2012) 807–824
40. Li, Y., Heflin, J.: Using reformulation trees to optimize queries over distributed heterogeneous sources. In: The Semantic Web–ISWC 2010. Springer (2010) 502–517
41. Li, Y.: A Federated Query Answering System for Semantic Web Data. PhD thesis, Lehigh University (2013)
42. Makris, K., Gioldasis, N., Bikakis, N., Christodoulakis, S.: Sparql rewriting for query mediation over mapped ontologies. Technical report, Tech. rep., Technical University of Crete (2010)
43. Makris, K., Gioldasis, N., Bikakis, N., Christodoulakis, S.: Ontology mapping and SPARQL rewriting for querying federated RDF data sources. In: On the Move to Meaningful Internet Systems, OTM 2010. Springer (2010) 1108–1117
44. OWL 2 Web Ontology Language Profiles (Second Edition). http://www.w3.org/TR/owl2-profiles
45. Quest. http://ontop.inf.unibz.it/?page_id=7
46. Rodrıguez-Muro, M., Calvanese, D.: Quest, an OWL 2 QL reasoner for ontology-based data access. OWLED 2012 (2012)
47. Nemirovski, G., Nolle, A., Sicilia, Á., Ballarini, I., Corado, V.: Data Integration Driven Ontology Design, Case Study Smart City. In: The Semantic Smart City Workshop (SemCity-13). (2013) accepted, to appear.
48. Shvaiko, P., Euzenat, J.: Ontology matching: state of the art and future challenges. **25** (2013) 158–176
49. Kiryakov, A., Damova, M.: Storing the semantic web: Repositories. Handbook of Semantic Web Technologies **1** (2011) 231–297
50. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three, AAAI Press (2011) 2656–2661
51. Kikot, S., Kontchakov, R., Podolskii, V., Zakharyaschev, M.: Long Rewritings, Short Rewritings. In: The 2012 Int. Workshop on Description Logics, CEUR Electronic Workshop Proceedings (2012) 235–245
52. Kikot, S., Kontchakov, R., Zakharyaschev, M.: On (in) tractability of OBDA with OWL 2 QL. In: Proc. of the 24th Int. Workshop on Description Logic. (2011) 224–234
53. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.U., Umbrich, J.: Data summaries for on-demand queries over linked data. In: Proceedings of the 19th international conference on World wide web, ACM (2010) 411–420
54. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD international conference on Management of data, ACM (1984) 47–57
55. Prasser, F., Kemper, A., Kuhn, K.A.: Efficient distributed query processing for autonomous RDF databases. In: Proceedings of the 15th International Conference on Extending Database Technology, ACM (2012) 372–383
56. SEMANCO. http://semanco-project.eu