Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2009) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

Applied Models and Knowledge Representation in Computational Science Workshop, KREAM
International Conference on Computational Science, ICCS 2012

Using a POWDER Triple Store for boosting the real-time performance of global agricultural data infrastructures

Pythagoras Karampiperis[a,*], Nikos Manouselis[b], Stasinos Konstantopoulos[a]

[a] *Institute of Informatics and Telecommunications, National Center for Scientific Research "Demokritos", Agia Paraskevi Attikis, P.O.Box 60228, 15310 Athens, Greece*

[b] *Agro-Know Technologies, Grammou 17 Str., Vrilissia Attikis, 15235 Athens, Greece*

Abstract

As the trend to open up data and provide them freely on the Internet intensifies, the opportunities to create added value by combining and cross-indexing heterogeneous data at a large scale increase. To seize these opportunities we need infrastructure that is not only efficient, real-time responsive and scalable but is also flexible and robust enough to welcome data in any schema and form and to transparently relegate and translate queries from a unifying end-point to the multitude of data services that make up the open data cloud. Transparent relegation and translation relies on detailed and accurate data summaries and other data source annotations, and with increased data volumes and heterogeneity managing these annotations, it becomes by itself a challenging data problem. In this position paper we discuss (a) how a scalable and robust semantic storage can be developed, using indexing algorithms that can take advantage of resource naming conventions and other natural groupings of URIs to compress data source annotations about extremely large datasets; and (b) how query decomposition, source selection, and distributed querying methods can be designed, that take advantage of such algorithms to implement a scalable and robust infrastructure for data service federation.

Keywords: Semantic Web; Metadata Publishing; Triple Store Compression; Resource Discovery

1. Introduction

During the last years, the trend to open up data and provide them freely on the Internet has intensified in volume as well as quality and value of the data made available. The linked data community has grasped the opportunity to combine, cross-reference, and analyse unprecedented volumes of high-quality data and to build innovative applications. This effort has caused a tremendous network effect, adding value and creating new opportunities for everybody, including the original data providers.

But most of the low-hanging fruit has been picked and it is time to move on to the next step, combining, cross-indexing and, in general, making the best out of all public data, regardless of their schema, size, and update rate; accepting that some schemas might be better suited to a given dataset and application and that there is no consensus about a "universal" schema or vocabulary for any given application, let alone for the Semantic Web and related

* Corresponding author. Tel.: +30-210-6503162; fax: +30-210-6532175.
E-mail address: pythk@iit.demokritos.gr.

initiatives such as the LOD cloud. In other words, we need infrastructure that besides being efficient, real-time responsive, and scalable is also flexible and robust enough to allow data providers to publish in the manner and form that best suits their processes and purposes and data consumers to query in the manner and form that best suits theirs.

This will be a decisive factor in maintaining the momentum of the linked open data movement by including in the cloud large, live, constantly updated datasets and streams that are published in formats that were not designed with linking across sources in mind. This will not only increase the value of all public data, but can also provide both the incentive and the opportunity to follow Semantic Web standards and linked data best practises for publishers that will not or cannot directly and immediately make this transition.

In order to achieve this ambitious vision and solve a difficult data management problem, the following key challenges should be addressed:

- Develop novel algorithms and methods for querying distributed triple stores, that can overcome the problems stemming from heterogeneity and from the fact that the distribution of data over nodes is not determined by the needs of better load balancing and more efficient resource discovery, but by data providers.
- Develop scalable and robust semantic indexing algorithms that can serve detailed and accurate data summaries and other data source annotations about extremely large datasets. Such annotations are crucial for distributed querying, as they support the decomposition of queries and the selection of the data sources which each query component will be directed to.
- Since it is, in the general case, not possible to align schemas and vocabularies so perfectly that there is no loss of information, investigate how to minimize losses and how to not accumulate them over successive schema translations.

Agricultural resource management is a good example of a real-world situation where data-intensive analysis needs to combine information from different, large-scale sources that are actively maintained in incompatible schemata: the agricultural domain includes various different topics with subjects varying from plant science and horticulture, to agricultural engineering, to agricultural economics. These different subjects are extensively researched by scientists all over the world, consuming as well as producing an enormous volume of data; agricultural scientists are inundated by an abundance of data as well as reported results relevant to their research as much as their colleagues from different disciplines.

To move towards these challenges, we propose to exploit the W3C POWDER protocol in order to maintain more detailed data summaries and data source annotations, and do at a larger scale than is possible today.

The paper is structured as follows: First, we discuss issues related with the implementation of a POWDER triple store. Then, we discuss the related technologies required to improve such implementations, which include distributed querying, approximate ontology alignment and query rewriting. Moreover, we present the proposed overall architectural solution, and discuss experimental results measuring the performance of POWDER triple stores. Finally, we discuss our findings and the conclusions that can be offered.

2. Implementing a POWDER Triple Store

POWDER is a Semantic Web technology that takes advantage of natural groupings of URIs in order to annotate all the resources in a regular expression-delineated sub-space of the URI space. The use cases for POWDER are centred on machine-readable trust marks, disambiguation of subject matter, declarations of accessibility compliance, mobile-friendliness, on-line safety, and so on. These all share the need to make statements about (parts of) Web sites, which is a more generally recurring need. POWDER allows making explicit the intention behind URI structure, which goes beyond exhaustively annotating all resources in a domain as it also represents the knowledge that not only currently known, but also unknown and future resources within a given URI space will have the properties implied by their position in the URI structure. For example, POWDER allows us to express the knowledge that any resource, currently existing or added in the future, under http://red.com has value "red" for the ex:colour property.

Compared to the ad-hoc formalisms and other standards fulfilling similar use cases such as robots.txt files, RFC 5785, and the PICS protocol, POWDER offers greater flexibility in defining URI spaces (regular expressions as opposed to URI prefixes) and has a rigorously defined semantics [1] that affords it a well-defined position the Semantic Web architecture and interoperability with RDF applications.

POWDER has seen various implementations applied to different applications and domains, including publishing trust marks conferred by third-party authorities to on-line medical content [2] and repository compression [2;3]. All these implementations share the property of having very limited interaction with non-POWDER semantic data as POWDER results are combined with other data following a layered inference approach where POWDER statements contribute to semantic inference but not vice versa.

Although implementing POWDER as a distinct layer is sanctioned by the formal semantics of POWDER statements, this direct implementation of POWDER semantics forces a choice between efficiency and compression. On the one hand, forward-chaining approaches either generate all POWDER-inferred triples and push those to the semantic inference layer above or implement a combined RDFS/POWDER inference engine [2] that, again, generates all POWDER and RDFS-inferred triples. Either way, any benefits POWDER can have to repository compression are lost since the POWDER-inferred triples are made explicit. On the other hand, backward-chaining approaches such as query rewriting achieve compression at the expense of efficiency. Query rewriting operates by transforming triple patterns involving POWDER-inferred predicates into equivalent FILTER clauses which apply the regular expression that the resource must match for the predication to hold. As a result, and as previously noted by Konstantopoulos and Archer [3], even the most restrictive POWDER statements can never be used to guard a query but can only be applied as tests over resources selected by previous triple patterns.

To fully realize the potential of POWDER we need to re-think some of the fundamental assumptions made when designing triple stores and databases in general. The BTree is the data structure that underlies modern triple stores and relational databases, coupled with hash tables that index the externally visible values against the internal node IDs. These data structures support the following strategy when searching for a triple pattern (or, in general, a tuple):

- The bound values in the triple are looked up in the hash table and replaced with internal node IDs
- The node IDs in the tuple are looked up in the BTree
- The unbound variables are bound by the values retrieved from the BTree

This strategy is very efficient when the variables in query patterns are bound to full URIs, but breaks down when retrieving POWDER annotations, as explained in the below sections.

2.1. Retrieving Resources by Property

Let us assume that we are looking for all resources that have a given property. Current approaches maintain two or more BTrees for looking up the properties of a given resource (spo indexing) or the resources that have a given property (pos indexing) using the strategy outlined above. But no index of nodes that correspond to specific resources or values can efficiently retrieve all resources that have a given POWDER-inferred property, without testing every single URI in the knowledge base against the regular expression.

One possibility would be to maintain in the structure all (relevant to POWDER statements) regular expressions a node's URI matches. Besides the large space usage (effectively annulling any compression benefits from POWDER) this would require prohibitive population costs for adding a POWDER statement to a large repository, since every single URI would have to be visited and checked against the new statement's regular expression. It becomes obvious that to efficiently implement POWDER processing we need to also be able to efficiently retrieve all URIs that match a regular expression.

2.2. Retrieving Properties of Resources

Let us now assume that we are looking for the filler of a POWDER-assigned property when the subject is bound to a URI. We look up the URI in the hash table, but we find no explicit triples in the tree. A possible solution would be to have regular expressions themselves be the subject in the tree, and have the hash table return not only the internal ID of the URI, but also the internal IDs of all matching regular expressions. This is sub-optimal because it (a) dramatically increases population costs (every single entry in the hash table needs to be revisited and updated when a new POWDER statement is added) and (b) increases querying time by causing unnecessary regular expression tests.

3. Source Selection and Distributed Querying

Repository federation and distributed querying are key technologies for efficient and scalable large-scale semantic repositories. The support for federated querying in most major repository implementations will be formalized in the upcoming SPARQL 1.1 specification. The new specification does not target federation that is transparent to the user, which is an open research question, but will include the SERVICE keyword through which query authors can specify which repositories to query about each triple pattern in the query.

Among the various approaches to optimizing distributed repositories, many target homogeneous databases where the same kind of information is stored across all nodes of the system using the same (or compatible) schema. Furthermore, such approaches typically require control over the way triples are hashed over the cluster, in order to optimize the distribution of triples in such a way that inter-node communication is minimized [4]. Although this level of control allows considerable optimizations, more dynamic solutions are needed as well as many real-world use cases: the ability to formally describe and take into account a prior and externally imposed data partitioning that the system does not control.

In the literature, several systems maintain indexes of the (kinds of) information stored at each source. Schema-level indexes are light-weight indexes of the properties and types (classes) that occur at each source [5]. Although efficient to maintain, such indexes are too coarse as they are missing instance-level information. So, for example, looking up repositories based on classes and properties from commonly used vocabularies (e.g., FOAF in general or AGROVOC in the agricultural domain) will return many false positives and thwart meaningful optimization.

Data summaries, on the other hand, combine schema and instance-level indexing to perform source selection. These stem from database concepts such as histograms, capturing statistics on selectivity and cardinality for the purpose of query optimization [6]. In order to index statistics about RDF triples, Q-Trees [7] combine the notion of histograms with that of R-Trees [8], multi-dimensional trees typically used to index spatial data. The core idea is that a hash function maps URIs and values to numerical "coordinates" that are used to find the node(s) in the Q-Tree corresponding to a (possibly only partially instantiated) triple pattern. Each such node aggregates statistics about all triples within its "bounding box"; this statistics is a ranked list of sources where data should be looked up.

The geometric metaphor employed by Q-Trees works well with heterogeneous data and is also robust to slightly outdated indexes [9]. However (a) does not allow for explicit declarations by repository maintainers regarding the kinds of data their repositories contain, but only relies on indexes incrementally built and maintained by successive queries; (b) fails take into account the semantic similarity between resources and relies instead on a purely syntactic mapping from URIs and value strings to numerical coordinates. In fact, the method is known to be very sensitive to the hashing function, with no universally good function found [7]; (c) fails to intelligently break up buckets when capacity is exceeded, relying on simple area-minimizing strategies

Query results from the end-points involved in a complex query are transformed back into the schema of the original query and joined. As interdependences between the sub-queries can degenerate this process into a situation where massive data volumes need to be copied to and processed by the results collector, the proposed solution builds upon methods for approximately joining distributed query results [4] and for distributed and approximate inference.

4. Approximate Ontology Alignment and Query Rewriting

The thorny issue of integrating heterogeneous information sources is open for the database community until today. Within the Semantic Web, and following the results of FP6-IST Knowledge Web [10], the problem of ontology alignment is defined as that of finding the correspondences between the entities (classes, properties, instances) of two ontologies. Each correspondence can be a relation between one entity from one ontology and one entity from another (1-1 correspondence) or between sets of entities (m-n correspondence); correspondences also carry a type (e.g. equivalent, subsumes, overlaps, etc.) and a degree of confidence.

4.1. Method Selection and Synthesis

When aligning heterogeneous information sources one has to cope with various phenomena that go beyond discovering the mapping between vocabulary terms, including expressivity differences (full ontologies vs. thesauri

or flat vocabularies of terms), granularity differences, etc. Furthermore, the process is affected by the correspondence types that are interesting to the application and the type of information made available to the alignment methods.

From the perspective of alignment methods, these adopt techniques from formal reasoning, graph analysis, machine learning, and language technology, and exploit different features of the ontologies including structural similarity between the graphs themselves, lexical similarity between the linguistic terms that lexicalize ontology entities, or semantic similarity discoveries via matching instances or the co-occurrence and frequencies of terms related to ontology entities. Selecting the best alignment method for a given pair of ontologies and target correspondence relations is done through an analytic hierarchy process [11], ad hoc rules [12], or measuring similarity in a representation designed to capture pertinent semantic and structure information [13].

Naturally, different methods exploit different types of information or different facets of the same information, consult (or not) different external resources, and target different types of correspondences. As a result, there is often no optimal method even for a given pair of ontologies; a synthesis, however, can make the best of multiple methods and producing an alignment that outperforms each of its individual constituents.

Many state of the art systems [14] combine method selection and synthesis into a biased synthesis approach where the correspondences discovered by multiple alignment methods are combined applying metrics of preference. Model-based approaches have individual methods provide results to a generic model that weights and integrates them using a specific computational model. Such a model is for example provided by a least square linear regression [15] or machine learning methods such as AdaBoost [16]. Recent model-based approaches view constituent methods as interacting agents, each responsible for using a specific alignment method to make alignment decisions. Synthesis is, then, viewed as the well-studied problem of maximizing social welfare, the sum of the utilities of the individual agents [17].

It is interesting to note that this latter method assures the consistency of the final result by taking into account the semantics of the schema language (e.g., OWL, RDFS) used by the aligned ontologies. This brings into focus the more general pattern of using meta-information about the ontologies under alignment in order to handle the syntactic and structural variation that can be encountered in a dynamic setting like the one examined in this paper aligning semantic resources often requires specific configuration and tuning for making them truly accessible and comparable and for setting appropriate exactness and accuracy expectations for the outcome of the process.

The idea is not new: as early as 2001 the FIPA Ontology Services Specification featured an Ontology Agent Ontology, a vocabulary that facilitated communication between ontology agents by providing metadata such as the knowledge representation schema which each ontology agent adhered to and W3C published the Web Service Definition Language for abstractly describing data and processing network services. More recently, the RDFStats Vocabulary [18] formalizes the publishing of histograms and other data summaries of RDF stores, but such information is also useful for alignment methods that exploit entity frequencies to discover mappings. The W3C Vocabulary of Interlinked Datasets (VoID) is designed with resource discovery and dataset archiving and cataloguing in mind and focuses on provenance information, but also defines terms for describing the links across datasets. Finally, the on-going W3C OntoLex activity is developing models for representing the linguistic knowledge (lexical, morphosyntactic, semantic, and pragmatic) required to realize ontological entities in different natural languages. Although OntoLex resources can be valuable for language technology-based alignment methods, the activity is not targeting alignment and does not intend to cover other aspects of ontology meta-information.

Although these vocabularies were originally developed for purposes other than support ontology alignment, they both establish frameworks and define vocabulary items that data providers can use to annotate their datasets for the purposes of configuring alignment.

However, no standardization initiative has established a unified and universally recognizable schema for ontology mediation.

4.2. Method Application and Evaluation

A reliable method for quantifying alignment quality can greatly improve performance, as it will allow source selection to prefer those data sources that can most accurately respond to the query. Alignment evaluation borrows from information retrieval the concepts of precision and recall, to propose their semantic variations [19] and also applies context-specific measures similar to the measures used in evaluating ontology learning methods [20].

Although alignment evaluation in general remains an open issue, given sufficient meta-information to properly tune and configure the system, a sufficient approximation can be computed for the purposes of source selection.

Furthermore, it is foreseen that alignment results are used not only to map queries to the vocabulary of the data source, but also to translate query results back to the schema of the original query. Although straight-forward for 1-1 mappings, this process gets dramatically convoluted for m-n mappings and perfect precision and recall is practically impossible as it often requires background knowledge that lies outside the formal conceptualization captured by the repository schemata. Recent work in the context of FP7-ICT SYNC3, views pattern translation as an inference process. This approach can contribute significant space and time optimization in storing and applying such translations patterns [21].

5. Proposed Architectural Solution

The proposed solution is to offer SPARQL endpoints that federate SPARQL endpoints over heterogeneous and diverse data sources, as depicted in Figure 1, which presents the proposed overall architecture.

The resource discovery and query decomposition component analyses SPARQL queries and uses metadata about the schema used and the instances stored in the various federated data stores in order to break up the original query into the optimal query fragments and decide where to forward each such fragment for execution. “Optimal” in this context involves a multitude of considerations, including minimizing the number of fragments (since joining the results carries considerable computational costs), schema proximity (minimizing the schema translation needed) and load balancing (preferring less used repositories).

The resource discovery and query decomposition mechanism relies on using POWDER to mass-annotate large sub-spaces of the URI space, allowing the system to take advantage of naming convention regularities to compress its indexes. POWDER can concisely annotate regular-expression delineated URI spaces with a single statement, affording efficiency and scalability in storing and maintaining data summaries. The decomposition rules of the system are refined during system operation by comparing the results that are returned against the system's expectations.

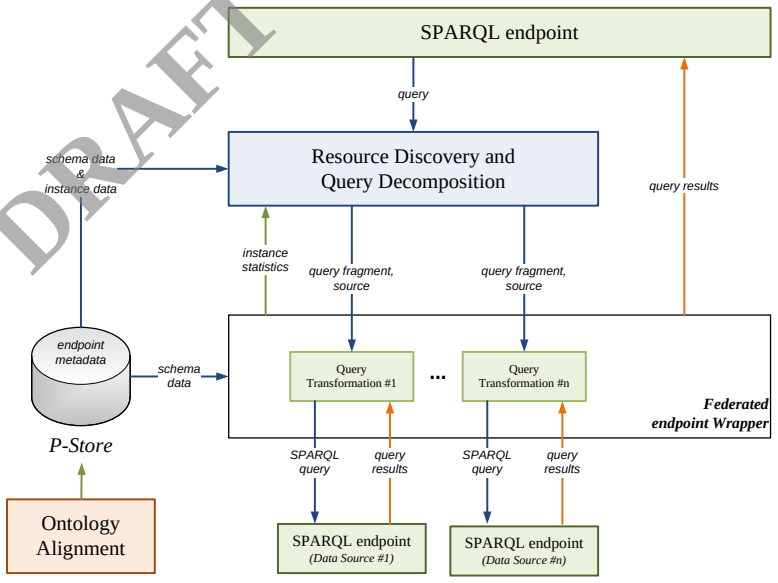

Fig. 1. Architecture of the proposed solution

The result of this process is a partitioning of the triple patterns in the original query where each fragment is annotated with an ordered list of data sources from the most likely to contain relevant data downwards. Resource discovery is by necessity an approximation, since completeness can only be guaranteed by querying all sources, but our method exhibits pay-as-you-go behaviour: the application posing the query can decide how much time it is

possible and worth it to wait to get more results, or set the minimum number of results required, or apply some other similar policy balancing between completeness and effort.

The federated end-point wrapper is responsible for managing the distributed querying and joining the results. Where necessary, ontology alignment results are used to transform the vocabulary of the query into the vocabulary of the data source. As schemas and vocabularies do not always align perfectly, due to overlaps between concepts, difference in the granularity of the conceptualization, etc., this is an approximate process involving computational intelligence rules that refine query terms to map them unto source vocabulary terms. These rules can exploit contextual information (e.g., previous queries by this user) in order to refine the query to the level needed by the source schema.

Query results are transformed back into the schema of the original query and joined by the federated end-point wrapper. As interdependences between the sub-queries can degenerate this process into a situation where massive data volumes need to be copied to and processed by the results collector, the system will develop methods for approximately joining distributed query results. We envisage a join operator that employs heuristics to exhibit pay-as-you-go behaviour, providing a first approximation with minimal usage of computational resources and iteratively refining it if more computation time and space are warranted by the application.

The system also foresees update and maintenance cycles, where new end-points are added to the federation or update the schema they employ or have accumulated considerable changes in the instances they hold. Schema metadata must be provided by the data provider when joining the federation, using authoring tools and tutorials produced by the project. Instance metadata may also be provided, but are also automatically maintained by the resource discovery module based on statistics extracted from query results. The rules and metadata used to achieve query transformation are derived by ontology alignment methods as well as human supervision.

6. Empirical experimentation of current POWDER implementations

Empirical experimentation on the space compression and the computation efficiency of current POWDER implementations has been carried out in the context of FP7-ICT SYNC3 [3] and has shown POWDER to achieve moderate compression rates (at the order of 20%) at no computational expense.SYNC3 extracts metadata about news events from on-line news content at a rate of 38Mtriples per month. The current system instance has been operating since April 2011 and has roughly reached 300Mtriple. It should be noted, however, that the SYNC3 repository deploys a POWDER implementation that only supports the constructs foreseen in the POWDER Recommendation, using regular expression matching to assign fixed property values to resources; and that the SYNC3 schema is only moderately amenable to POWDER compression.

Table. 1. Two LUBM Query Examples

LUBM Query A	**LUBM Query B**

SELECT ?X WHERE {
?X rdf:type
lubm:UndergraduateStudent .
}
	SELECT ?X ?Y WHERE {
	?X rdf:type lubm:Student .
	?Y rdf:type lubm:Course .
	data:Department0.University0.edu/AssociateProfessor0
	lubm:teacherOf ?Y .
	?X lubm:takesCourse> ?Y .
	}

To better understand the strengths and weaknesses of current POWDER implementations we have conducted further experiments with a 500 Megatriple repository generated using the Lehigh University Benchmark (LUBM). Two variations of this dataset were prepared, one containing all RDF data and one where rdf:type information that can be inferred from the URI was removed; for example:

```
data:UnderGraduateStudent0 rdf:type lubm:UnderGraduateStudent .
data:Department0.University0.edu/AssociateProfessor0 rdf:type lubm: AssociateProfessor .
```

Queries on this latter repository were executed via the backward-chaining SYNC3 POWDER implementation that applies backward-chaining inference to rewrite POWDER-inferable triple patterns in queries into equivalent FILTER clauses applying the regular expressions specified by the POWDER document that describes the data.

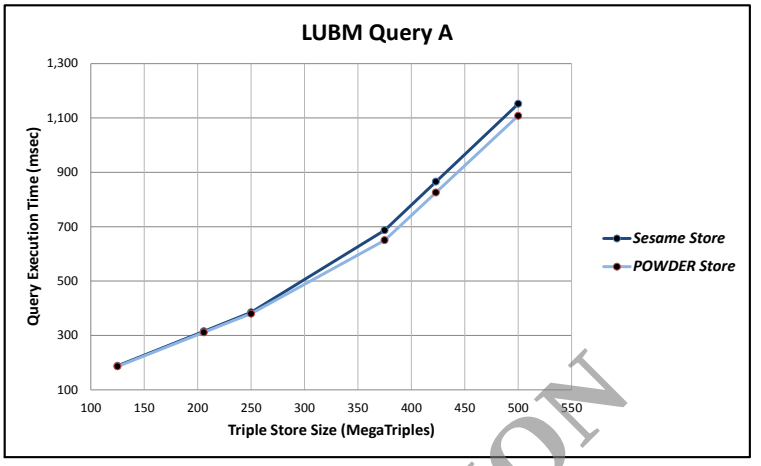

Fig. 2. Performance in query example A

Results for the two queries on Table 1 are shown in Figures 2 and 3. The data sizes quoted in these figures assume that the POWDER repository holds data summaries at the, relatively dense, ratio of 1:100. That is, 1 triple with meta-information is held in the query decomposition and data selection node for every 100 triples in the distributed data sources. The times quoted reflect the increase in the time it takes to pre-process queries before the can be distributed as the end-point federation grows larger.

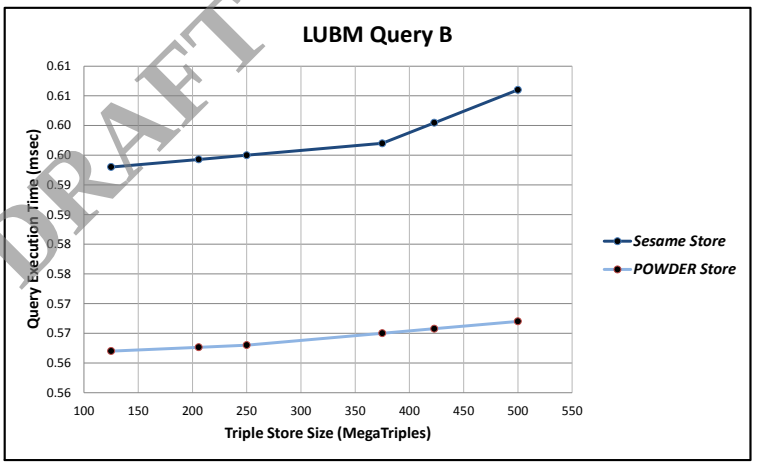

Fig. 3. Performance in query example B

Query A is a high-throughput query that yields a number of tuples directly proportional to repository size; POWDER has no practical effect on these types of queries, especially since there is no non-POWDER pattern to guard the query. Query B is a query about a specific resource with a fixed throughput regardless of repository size; POWDER has a more profound effect since two out of the three triple patterns in the query (the two rdf:type patterns) do not result in a lookup in the index but are satisfied by matching the bindings of the ?X and ?Y variables against regular expressions; the existence of very restrictive guards (patterns with grounded resources) maximizes the benefit from POWDER.

Naturally, since the computation needed to match the regular expression does not depend on the size of the

repository but only on the number and complexity of POWDER descriptions, POWDER vs. vanilla performance over Query B diverges as the repository grows bigger. In other words, POWDER achieves scalability by loosening the association between the time it takes to annotate resources ?X and ?Y from the size of the repository.

Two factors should be noted when projecting the efficiency of a naïve POWDER store:

- The naïve POWDER store will dramatically improve performance on Query A-type queries.
- Neither the SYNC3 not the LUBM schema, are well-suited for drastic compression under the currently implemented expressivity.

Especially with respect to this latter point, the naïve POWDER store will allow more extensive use of regular expressions by using them to compute property values instead of only permitting regular expression recognizers. So, for example, the LUBM URI convention would be better exploited inferring further properties besides the type, such as the university and department where they are employed, from resources such as:

```
data:Department0.University0.edu/AssociateProfessor0
```

Such a mapping formalism will be particularly useful in source selection and ontology alignment, and especially for instance-based meta-information. Consider, for example, the following URIs:

```
http://www.geonames.org/264371/athens.html
http://dbpedia.org/page/Athens
```

and the opportunity and extended POWDER implementation offers to succinctly describe the first guess one would make to generate the latter from the former. Such a description can reduce great volumes of triples of mapping knowledge to a single statement and a handful of exceptions, such as

```
http://www.geonames.org/ 4180386/athens.html
http://dbpedia.org/page/Athens,_Georgia
```

It is, thus, expected that the compression achieved can reach the order of 90% with querying time growing at a sub-linear rate against total data volume, resembling Figure 3 rather than Figure 2.

7. Conclusions & Future Work

In this position paper we discuss (a) how a scalable and robust semantic storage can be developed, using indexing algorithms that can take advantage of resource naming conventions and other natural groupings of URIs to compress data source annotations about extremely large datasets; and (b) how query decomposition, source selection, and distributed querying methods can be designed, that take advantage of such algorithms to implement a scalable and robust infrastructure for data service federation.

Future work includes the development of a distributed infrastructure layer on top of existing data repositories and networks that will support the interoperable and transparent application of data-intensive techniques over heterogeneous data sources. This infrastructure will integrate:

- Novel indexing algorithms that support the efficient storage and retrieval of data summaries that concisely describe instance-level metadata about the different sources federated under the infrastructure. It is envisaged that such algorithms will support advanced source selection methods over distributed databases, affording scalability by disassociating or sub-linearly associating the time it takes to decide which repository to ask for a given piece of information from the number and size of the repositories that the distributed system comprises.
- An extension of state-of-the-art query decomposition and rewriting methods that will enable complex queries in one schema to be broken down into sub-queries, each in a (possibly) different schema. In synergy with the distributed source selection mechanisms, this will allow queries in any schema to be executed at all and only those repositories that might hold relevant information, regardless of the schema these repositories use.
- The integration of a variety of state-of-the-art schema alignment methods under a novel architecture for the prior selection of the most appropriate method or methods for a given schema pair, the synthesis of multiple methods into a unified alignment, and the posterior evaluation of alignment quality. Alignment results will be used at querying time to rewrite queries (or query fragments) from the query schema into the source schema and results from the source schema back into the query schema.

Acknowledgements

The work of Nikos Manouselis has been supported by the EU project agINFRA of the FP7 Research Infrastructures, Capacities Programme (http://aginfra.eu).

References

[1] S. Konstantopoulos and P. Archer, "Protocol for Web Description Resources (POWDER): Formal Semantics", W3C Recommendation, 1 September 2009.

[2] S. Konstantopoulos, P. Archer, P. Karampiperis, and V. Karkaletsis, "The POWDER protocol as infrastructure for serving and compressing semantic data", International Journal of Metadata, Semantics, and Ontologies. Accepted to appear in 2012.

[3] S. Konstantopoulos and P. Archer, "POWDER and the multi-million triple store". In: Proceedings of the 3rd International Workshop on Semantic Web Information Management (SWIM 2011), ACM SIGMOD/PODS 2011, Athens, Greece, 12-16 June 2011.

[4] J. Huang, D.J. Abadi, and K. Ren, "Scalable SPARQL querying of large RDF graphs". In: Proceedings of the VLDB Endowment, Vol. 4(11), 2011.

[5] H. Stuckenschmidt, R. Vdovjak, G.-J. Houben, and J. Broekstra, "Index structures and algorithms for querying distributed RDF repositories". In: Proceedings of the 13th International World Wide Web Conference (WWW 2004), New York, USA, 17-22 May 2004.

[6] Y. Ioannidis, "The history of histograms (abridged)", In: Proceedings of the 29th International Conference on Very Large Databases (VLDB 2003), Berlin, Germany, 9-12 September 2003. Ten-Year Best Paper Award.

[7] A. Harth, K. Hose, M. Karnstedt, A. Polleres, K.-U. Sattler, J. Umbrich, "Data summaries for on-demand queries over Linked Data", In: Proceedings of the 19th International World Wide Web Conference (WWW 2010), Raleigh, NC, USA, 26-30 April 2010.

[8] A. Guttman, R-Trees, "A dynamic index structure for spatial indexing", In: Proceedings of the Annual Meeting of ACM SIG on the Management of Data (SIGMOD '84), Boston, MA, USA, 18-21 June 1984.

[9] K. Hose, D. Klan, and K.-U. Sattler, "Distributed data summaries for approximate query processing in PDMS". In: Proceedings of the 10th International Database Engineering and Applications Symposium (IDEAS '06), Delhi, India, 11-14 December 2006.

[10] P. Bouquet, J. Euzenat, E. Franconi, L. Serafini, G. Stamou, and S. Tessaris, "Specification of a common framework for characterizing alignment", Knowledge Web Deliverable 2.2.1, 2004.

[11] M. Mochol, A. Jentzsch, and J. Euzenat, "Applying an analytic method for matching approach selection", In: Proceedings of the 1st International Workshop on Ontology Matching (OM 2006), held at the 5th International Semantic Web Conference (ISWC2006), Athens, GA, USA, November 2006, pp. 37–48.

[12] M. Mochol and A. Jentzsch, "Towards a rule-based matcher selection", In: Proceedings of the 16th International Conference on Knowledge Engineering: Practice and Patterns (EKAW), 2008, pp. 109–119.

[13] A. Algergawy, R. Nayak, N. Siegmund, V. Koppen, and G. Saake, "Combining schema and level-based matching for web service discovery", In: Proceedings of the 10th International Conference on Web Engineering (ICWE), 2010, pp. 114–128.

[14] P. Shvaiko, J.Euzenat, "Ontology Matching: State of the Art and Future Challenges", IEEE Transactions on Knowledge and Data Engineering 99, 2011.

[15] A. Doan, P. Domingos, and A. Halevy, "Reconciling schemas of disparate data sources: A machine-learning approach", In: Proceedings of the 20th International Conference on Management of Data (SIGMOD), 2001, pp. 509–520.

[16] A. Marie and A. Gal, "Boosting schema matchers", In: Proceedings of the 16th International Conference on Cooperative Information Systems (CoopIS), 2008.

[17] V. Spiliopoulos and G. A. Vouros, "Synthesizing ontology alignment methods using the max-sum algorithm", IEEE Transactions on Knowledge and Data Engineering, 2011.

[18] A. Langegger and W. Woess, "RDFStats: An extendible RDF statistics generator and library", In: Proceedings of the 8th International Workshop on Web Semantics, held in conjunction with the 20th International Conference on Database and Expert Systems (DEXA 2009), Linz, Austria, 31 August – 4 September 2009.

[19] D. Fleischhacker and H. Stuckenschmidt, "A practical implementation of semantic precision and recall", In: Proceedings of the 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), 2010, pp. 986–991.

[20] E. Zavitsanos, G. Paliouras, and G. A. Vouros, "Gold standard evaluation of ontology learning methods through ontology transformation and alignment", IEEE Transactions on Knowledge and Data Engineering 23(11), November 2011, pp. 1635–1648.

[21] S. Konstantopoulos, "Using on-the-fly pattern transformation to serve multi-faceted event metadata", In: Proceedings of the Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2011), held at the 10th International Semantic Web Conference (ISWC 2011), Bonn, Germany, 23-27 October 2011.