A Semantic Web Middleware for Virtual Data Integration on the Web

Andreas Langegger, Wolfram Wöß, and Martin Blöchl

Institute of Applied Knowledge Processing
Johannes Kepler University Linz
Altenberger Straße 69, 4040 Linz, Austria
{ al | martin.bloechl | wolfram.woess } @jku.at

Abstract. In this contribution a system is presented, which provides access to distributed data sources using Semantic Web technology. While it was primarily designed for data sharing and scientific collaboration, it is regarded as a base technology useful for many other Semantic Web applications. The proposed system allows to retrieve data using SPARQL queries, data sources can register and abandon freely, and all RDF Schema or OWL vocabularies can be used to describe their data, as long as they are accessible on the Web. Data heterogeneity is addressed by RDF-wrappers like D2R-Server placed on top of local information systems. A query does not directly refer to actual endpoints, instead it contains graph patterns adhering to a virtual data set. A mediator finally pulls and joins RDF data from different endpoints providing a transparent on-the-fly view to the end-user.
The SPARQL protocol has been defined to enable systematic data access to remote endpoints. However, remote SPARQL queries require the explicit notion of endpoint URIs. The presented system allows users to execute queries without the need to specify target endpoints. Additionally, it is possible to execute join and union operations across different remote endpoints. The optimization of such distributed operations is a key factor concerning the performance of the overall system. Therefore, proven concepts from database research can be applied.

1 Introduction

One of the best use cases for Semantic Web technology is probably large-scale data integration across institutional and national boundaries. Compared to traditional approaches based on relational database systems, there are several aspects of the Semantic Web, which makes it well suited for the integration of data from globally distributed, heterogeneous, and autonomous data sources. In short, these are: a simple but powerful and extensible data model which is the Resource Description Framework (RDF), URIs (or IRIs) used for global naming, and the possibility of reasoning based on Description Logic.

In this paper a system is presented which is developed primarily for sharing data in scientific communities. More precisely, the system is developed as part of the Austrian Grid project to enable transparent access to distributed data for scientific collaboration. The main project is called *Semantic Data Access Middleware for Grids* (G-SDAM)

[16]. Because of the generic architecture of the mediator component, which is responsible for processing queries, and because of its supposed relevance for the Semantic Web community, it has been detached from the rest of the Grid middleware. It is expected that other Semantic Web applications will benefit from this *Semantic Web Integrator and Query Engine* (SemWIQ). The system cannot only be used for other data integration applications such as library repositories, directories, or various scientific archives and knowledge bases, it could also be used to complement Semantic Web search engines and Linked Data browsers (explained in Section 6).

The architecture of the data integration system is based on the following findings: scientific data is usually structured[1], regional or globally distributed, stored in heterogeneous formats, and sometimes access is restricted to authorized people. To provide a transparent access to such kinds of data, a mediator-wrapper architecture is commonly used [29]. It basically consists of a mediator which is accepting queries from clients and then collects data translated by a number of wrappers attached to local data sources. These wrappers use mappings to translate data from the underlying information systems into a common schema which can be processed by the mediator. In the case of virtual data integration, mappings are used for on-the-fly translation of data during query processing. In the next section some related work will be discussed.

The remainder of the paper is structured as follows. After the related work section, the concept of the system and its architecture are described (Section 3). Details about query federation and the implementation of the mediator are presented (Section 4). A sample scenario with sample queries and results are presented (Section 5). Optimization concepts for distributed SPARQL queries and future work are discussed (Section 6) and finally, the conclusion can be found in Section 7.

2 Related Work

Related work can be divided into four main categories: (a) schema integration using ontologies, (b) schema mapping, and translating source data to RDF, (c) distributed query processing, and (d) similar projects addressing ontology-based data integration.

While schema integration is not a new topic in general (an overview can be found in [4]), ontology matching and alignment is rather new[2] [25]. The presented system uses OWL DL ontologies for the global data model and an extended version of SPARQL for processing queries. When integrating data sources of a specific (scientific) domain, it is required to create global ontologies that are expressive enough to describe all data that will be provided. Because data is usually not stored as RDF graphs originally, schema integration has to be done over arbitrary data models. This is a difficult task which is related to a fairly new discipline called *model management* [18]. On the other hand, ontologies have already been developed over the past years for several domains (e.g. medicine, biology, chemistry, environmental science). These can be reused and

[1] Apart from data which is shared before being analyzed, as for instance data collected at CERN which is distributed across Europe. But this is for scalability reasons, there is no data integration taking place.

[2] Starting with 2004, there is an annual workshop as part of the International Semantic Web Conference (ISWC): `http://oaei.ontologymatching.org`.

combined freely because of the modular nature of RDF which is based on namespaces identified by globally unique IRIs. Thus, the current process is finding associations between concepts of local data models and existing ontologies. For this process tools can help but for the creation of meaningful mappings experts will be needed.

Concerning the second topic (b), only mapping frameworks that allow manual or (semi-)automatic mapping from arbitrary data models to RDF are relevant. And since SPARQL is used for global queries at the mediator, these mappings have to support on-the-fly data access based on SPARQL. There are a few frameworks that support this [21, 22, 7][3]. Because the mapping language used by D2R-Server [7] is most powerful, it was chosen as a wrapper for relational database systems. For other information systems or protocols like LDAP, FTP, IMAP, etc. or CSV files and spreadsheets stored in file systems, it is possible to adapt the D2R wrapping engine. This has successfully been done for the library protocol standard OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) in the context of a library integration system developed at the University of Vienna [12].

A powerful capability of the presented system is the possibility to execute binary operations across distributed data sources. These operations are involved when matching multiple basic graph patterns, optional patterns, or alternatives. It is possible because the system is completely based on a pipelined query processing workflow. However, optimization of distributed query plans is a pre-condition in order to supply results as fast as possible. It seems that the discussion about federating SPARQL queries and query optimization in general is gaining importance in the community. Because shipping data over the internet is more expensive than most of the local operations, other policies and algorithms have to be used than for local queries. In [23] blocking of remote sub-queries has been described reflecting a common approach in distributed query processing which is called row blocking [9]. The re-ordering of binary operations (especially joins) is one of the most complex tasks in a distributed scenario. Some early conceptual ideas have been presented in [15], but further research and experiments will be required (Section 6).

Lessons learned from the development of the D2R wrapper showed that people are demanding for real data integration: “Mapping to RDF is not enough” [6]. Recently, several approaches addressing ontology-based data integration have been proposed. Some of them were initiated in an inter-disciplinary setting (e.g. biomedical [20]). Because of the large number of application scenarios related work will concentrate on implementations and systems. In [24] a concept-based data integration system is described which was developed to integrate data about cultural assets. Although RDF ontologies are used to describe these assets, query processing is based on XML and a special query language called CQuery. Web services are used as a common gateway between the mediator and data sources and XSLT is used to map XML data to the global structure. The authors also presented several optimization techniques like push-down of selections or cashing of results at the mediator. Two other projects based on RDF and SPARQL are *FeDeRate* [22] and *DARQ* [3]. The first one, FeDeRate, just adds support to SPARQL for remote sub-queries using named graph patterns. Thus, it can be seen

[3] A more comprehensive list is collected at `http://esw.w3.org/topic/RdfAndSql(Dec10, 2007).`

as a multi-database language because endpoints have to be specified explicitly. DARQ, which is an acronym for *Distributed ARQ*, provides access to multiple, distributed service endpoints and is probably related most closely to the presented system. However, there are some important differences. Setting up DARQ requires the user to explicitly supply a configuration file which includes endpoint descriptions. These descriptions include: capabilities in the form of lists of RDF property IRIs which are used at the endpoint for data descriptions, cardinalities for instances, and selectivities. But these statistics are no longer representative when the corresponding data is changed. The system presented here uses a concept-based approach based on DL ontologies, however the idea is similar: data source selection is based on type information instead of RDF properties only. Furthermore, a dynamic catalog is used storing meta data and statistics about available endpoints. These statistics are automatically gathered by a monitoring service. It has to be said that DARQ was only the prototype of a rather small project at HP Labs which unfortunately has not been continued yet.

Virtuoso [21], a comprehensive data integration software developed by OpenLink Software, is also capable of processing distributed queries. Because Virtuoso is also a native quad store, the strength of this software is its scalability and performance. Beside the commercial edition, there is also an open source version available. A relatively new application also provided by OpenLink is the *OpenLink Data Spaces* platform, which is promoted as being able to integrate numerous heterogeneous data from distributed endpoints. Finally, there is also a commercial-only software package, called *Semantic Discovery System* [8], claiming to be able to integrate all kinds of data from distributed sources based on SPARQL with optimal performance. Because no resource materials could be found about the internals the software has not been tested further.

3 Concept and Architecture

For the implementation of SemWIQ the Jena2 RDF library developed at HP Labs [14] is used. Since the system is based on the mediator-wrapper approach [29], its architecture depicted in Fig. 1 looks similar to other mediator-based systems. Clients establish a connection to the mediator and request data by submitting SPARQL queries (1). Patterns in such *global* queries adhere to a virtual graph which refers to classes and properties from arbitrary RDFS or OWL vocabularies. As long as these vocabularies are accessible on the Web according to the *Best Practice Recipes for Publishing RDF Vocabularies* [19], they can be used by data sources to describe provided data. The parser (2), which is part of Jena (ARQ2), calculates a canonical query plan which is modified by the federator/optimizer component (3). Query federation and optimization is tightly coupled and will be described in Section 4. The federator analyzes the query and scans the catalog for relevant registered data sources. The output of the federator/optimizer is an optimized global query plan which is forwarded to the query execution engine (4). The query execution engine processes the global plan which includes remote sub-plans executed at wrapper endpoints (5). Currently, the SPARQL protocol is used between mediator and wrapper endpoints and sub-plans are therefore serialized back into lexical queries. This is a first approach, but it is limited and does not allow for sophisticated optimization strategies as will be discussed later in Section 6.

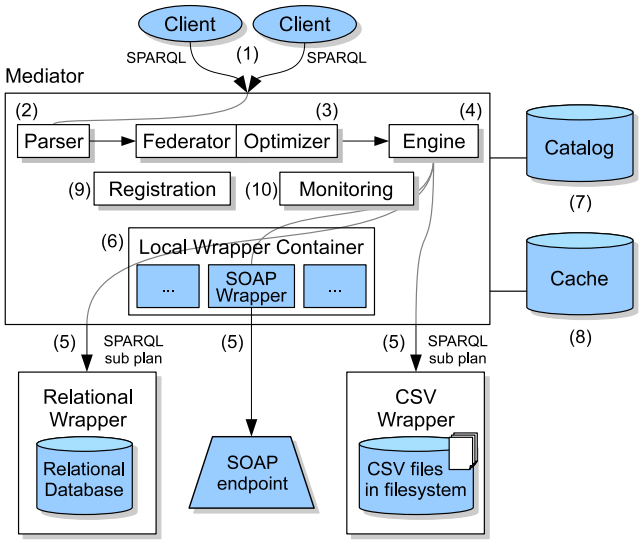

Fig. 1. Mediator-Wrapper architecture of SemWIQ.

Any data source must use a SPARQL-capable wrapper to provide local data unless it is a native RDF data source supporting the SPARQL protocol. As explained earlier there are several wrappers available for relational database systems of which D2R has been chosen because of its powerful mapping language. Usually, a wrapper is placed as close as possible to the local information system to be able to benefit from local query processing capabilities. However, it may occur that a data source cannot be extended by a wrapper (e.g. web service endpoints or generally, when there is no control over the remote system). For such cases, the mediator provides a local wrapper container (6). A wrapper inside the container may process a sub-plan and issue native access operations to the actual remote endpoint. The catalog (7) stores descriptions and statistics about registered data sources and a local RDF cache (8) will be used in future to cache triples for global join operations and recurring queries. The registration component is currently a simple REST-based service. A data source can register itself at the mediator by sending a HTTP POST request with an RDF document attached. It specifies the endpoint URI and meta data as for example the providing *Virtual Organization* (VO) and a contact person[4]. De-registration is currently not implemented. If the endpoint becomes unavailable, it is automatically removed from the catalog so it has to register again. The extension of the registration service is one part of future work. The monitoring component (9) is periodically fetching statistics from registered data sources.

[4] G-SDAM, mentioned in the introduction, uses the *Grid Security Infrastructure* (GSI) to authenticate requests and Virtual Organizations. Within SemWIQ virtually anybody can register a data source.

3.1 Catalog and Monitoring

The catalog is implemented as a fast in-memory Jena graph. It is currently persisted into a file when the mediator shuts down. However, since the Jena assembler API is used, it is just a matter of changing a configuration file to persist the whole catalog into a database. Beside the endpoint URI and description, it stores statistics gathered by the monitoring component. Because the optimization concepts explained later are not fully implemented in the current prototype, the statistics are rather simple. They are also used for data source selection by the federator. The currently used catalog vocabulary is described by `http://semwiq.faw.uni-linz.ac.at/core/2007-10-24/catalog.owl`.

The monitoring service is designed for simplicity and easy adoption. No configuration is required at remote data sources. They just need to register and the mediator will do the rest. It fetches data using an extended SPARQL syntax implemented in ARQ2 which allows for aggregate queries. The following pseudo-algorithm and queries are used to fetch a list of classes and the number of instances a data sources provides for each class as well as a list of properties and their occurrences:

```
foreach (DataSource as ds) {
  SELECT DISTINCT ?c WHERE { [] a ?c }
  foreach (solution mapping for ?c as c)
    addConceptCount(ds, c, (SELECT COUNT (*) WHERE { ?s a <c> }))

  SELECT DISTINCT ?p WHERE { [] ?p [] }
  foreach (solution mapping for ?p as p)
    addPropertyCount(ds, p, (SELECT COUNT(*) WHERE { ?s <p> ?o }))
}
```

The `COUNT(*)` statement is currently not part of the SPARQL recommendation and thus, there are a few different proprietary syntaxes around for aggregate functions. It would be better to use distinct counting, however this would break compatibility with Virtuoso and also DBpedia [2] as a consequence.

3.2 Wrapping Data to RDF

Most structured data is currently stored in relational database systems (RDBMS). However, especially in the scientific community file-based data formats are also very popular because sharing and interchange is often easier when having a file. While newer file-based data formats are often based on XML, there are several legacy formats around like FITS for instance, the *Flexible Image Transport System* endorsed by NASA and the International Astronomical Union. It can be compared to JPEG files including EXIF data. In terms of a RDBMS, this would be a record with several fields next to the BLOB or file path/URI where the image is stored. Because often tools already exist which enable the import from legacy data like FITS files into a RDBMS, a wrapper for these information systems is most important. D2R-Server [7] has been chosen because it uses a powerful mapping language. Later on, additional wrappers will be added for other information systems, protocols like LDAP, FTP, IMAP, etc. and CSV files or spreadsheets stored in file systems. D2R-Server also provides a good basis for the implementation of further wrappers.

The mapping language used by D2R can be seen as a global-as-view approach. Starting with global classes and properties defined using RDF Schema or OWL, a view is defined which maps tables and attributes from the database to the global concepts. Regarding query optimization, D2R relies on the optimizer of the local database management system. At the scale of the complete mediator-based data integration system, the wrapper more or less breaks the query processing pipeline. The optimizer at the mediator can currently not take into account special indices of local database systems. To benefit from these – which would decrease response time and lower memory consumption – it will be required to introduce a special protocol instead of the SPARQL protocol which is able to communicate capabilities and estimated costs during plan optimization. Functional mappings as well as translation tables are both supported by D2R. Scientific data sets often use varying scales and metrics. A functional mapping can transform data accordingly. A translation table can be used for varying nominal scales or naming schemes.

3.3 Vocabularies

Depending on the data an endpoint provides, different vocabularies to describe them may be used. There are many vocabularies around created for social software like *Friend-of-a-Friend* (foaf), *Description-of-a-Project* (doap), *Semantically-Interlinked Online Communities* (sioc), etc. and also several scientific communities have created more or less suitable ontologies which can be used to describe data. If there are no applicable vocabularies available for a specific domain, usually a small group of people will introduce a new one and publish them according to [19]. For instance, an ontology for solar observation which is used by the prototype scenario developed together with the Kanzelhöhe Solar Observatory.

A major design goal is to remain flexible and keep the setup process as simple as possible. Initially it was assumed to introduce a collaborative management system for globally used ontologies including versioning support. However, finding a consensus on collaboratively developed ontologies is a difficult, time-consuming, and sometimes frustrating process. It is often better to design vocabularies bottom-up than rounding up as many people as possible and following a top-down approach. Now, everybody can re-use and publish new vocabularies as long as they are accessible on the Web. The registered SPARQL endpoint will be queried by the monitoring service as described and automatically fetch required vocabularies from the Web.

3.4 Authentication and Data Provenance

Scientific data is often only shared inside a community where people know each other. To restrict access to a data source the Grid-enabled middleware G-SDAM uses the *Grid Security Infrastructure*. It is based on a PKI (*Private Key Infrastructure*) and each person using the service requires a Grid certificate issued by a central or regional authority. Data sources are usually provided by *Virtual Organizations*, a concept introduced by the Grid community [11]. For granting access on a data source to specific persons, tools supplied by current Grid middleware like Globus Toolkit or gLite can be used. Because

there are also components available for billing of used Grid resources, it may be possible in future to buy and sell scientific data over G-SDAM. Regarding the SemWIQ project, there is currently no support for fine-grained access control. Only the mediator may be secured by SSL and HTTP Basic Authentication which is actually up to the system administrator.

Another important aspect when sharing scientific data – which also may become more and more an issue when the Semantic Web takes off – is data provenance. For any data integration software, it is very important to know where data is coming from. Within SemWIQ this can be achieved by a convention. Upon registration the data source has to supply an instance of type `cat:DataSource`[5]. This instance must be identified by a fully-qualified IRI representing the data source. For each instance returned by the data source, a link to this IRI must be created using the `cat:origin`-property. The mediator will always bind this value to the magic variable `?_origin`. If this variable is added in the projection list, it will appear in the result set. If it is not specified or if the asterisk wildcard is used, it will not occur in the result set.

4 Federating SPARQL Queries

With the SPARQL Working Draft of March 2007 a definition of SPARQL [28, Section 12] was added which introduces a common algebra and semantics for query processing. Query plans can now be written using a prefix syntax similar to that of LISP. In this section the federator will be described. Since the mediator is based on Jena, it also uses ARQ for parsing global SPARQL queries. The idea of concept-based data integration is that every data item has at least one asserted type. For a global query, the mediator requires type information for each subject variable in order to be able to retrieve instances. This implies that there are several restrictions to standard SPARQL:

- All subjects must be variables and for each subject variable its type must be explicitly or implicitly (through DL constraints) defined. A BGP like {`?s :p ?o`} is not allowed unless there is another triple telling the type for `?s`. For instance, the BGP {`?s :p ?o ; rdf:type <some-type>`} is valid. In a future version, when `DESCRIBE`-queries are supported, it may become valid to constraint the subject term to an IRI.
- For virtual data integration it is not required to have multiple graphs. A query may only contain the default graph pattern. Furthermore, federation is done by the mediator and not explicitly by the user (this can be done with *FeDeRate* [22]).
- Currently only `SELECT`-queries are supported, but support for `DESCRIBE` is planned. Supporting `DESCRIBE` may be useful to get further information to an already known record, however the mediator has to go through all data sources.

All other features of SPARQL like matching group graph patterns, optional graph patterns, alternative graph patterns, filters, and solution modifiers are supported but they still require further optimization. To demonstrate the federation algorithm, Query

[5] The prefix `cat` is generally used for the namespace `http://semwiq.faw.uni-linz.ac.at/core/2007-10-24/catalog.owl#`

2 shown in Fig. 2 is taken as an example. Firstly, ARQ parses the query string and generates the canonical plan:

```
(project (?dt ?groups ?spots ?r ?fn ?ln)
  (filter (&& (= ?fn "Wolfgang") (= ?ln "Otruba"))
    (BGP
      (triple ?s rdf:type sobs:SunspotRelativeNumbers)
      (triple ?s sobs:dateTime ?dt)
      (triple ?s sobs:groups ?groups)
      (triple ?s sobs:spots ?spots)
      (triple ?s sobs:rValue ?r)
      (triple ?s obs:byObserver ?obs)
      (triple ?obs rdf:type obs:Observer)
      (triple ?obs person:firstName ?fn)
      (triple ?obs person:lastName ?ln)
    )))
```

For better readability namespace prefixes are still used for printing plans. Next, the federator transforms this plan according to the following Java-like algorithm:

```
visit(opBGP) by visiting plan bottom-up {
  Hashtable<Var, BasicPattern> sg = createSubjectGroups(opBGP); // group by subjs
  Op prev = null;

  // iterate over subject groups
  Iterator<Node> i = sg.keySet().iterator();
  while (i.hasNext()) {
    Node subj = i.next();
    String type = getType(sg, subj); //+does static caching of already known types
    OpBGP newBGP = new OpBGP((BasicPattern)sg.get(subj));
    Op prevU = null;

    // look for sites storing instances of determined type...
    Iterator<DataSource> dit = catalog.getAvailable(type);
    while(dit.hasNext()) {
      DataSource ds = dit.next();
         Op newDS = new OpService(Node.createURI(
              ds.getServiceEndpoint()), newBGP.copy());
      prevU = (prevU==null) ? newDS : new OpUnion(prevU, newDS);
    }
    prev = (prev==null) ? prevU : OpJoin.create(prev, prevU);
  }
  if (prev == null) return opBGP; // no data sources found
  else return prev;
}
```

Because there are two registered data sources providing instances of `obs:Observer` and `sobs:SunspotRelativeNumbers` the resulting (un-optimized) query plan is:

```
(project (?dt ?groups ?spots ?r ?fn ?ln)
  (filter (&& (= ?fn "Wolfgang") (= ?ln "Otruba"))
    (join
      (union
        (service <http://keas.kso.ac.at:8002/sparql>
            (BGP
              (triple ?s rdf:type sobs:SunspotRelativeNumbers)
              (triple ?s sobs:dateTime ?dt)
              (triple ?s sobs:groups ?groups)
              (triple ?s sobs:spots ?spots)
              (triple ?s sobs:rValue ?r)
              (triple ?s obs:byObserver ?obs)
            ))
          (service <http://solarscience.msfc.nasa.gov:8004/sparql>
```

```
    (BGP
      (triple ?s rdf:type sobs:SunspotRelativeNumbers)
      (triple ?s sobs:dateTime ?dt)
      (triple ?s sobs:groups ?groups)
      (triple ?s sobs:spots ?spots)
      (triple ?s sobs:rValue ?r)
      (triple ?s obs:byObserver ?obs)
    ))
)
(union
  (service <http://keas.kso.ac.at:8002/sparql>
      (BGP
        (triple ?obs rdf:type obs:Observer)
        (triple ?obs person:firstName ?fn)
        (triple ?obs person:lastName ?ln)
      ))
    (service <http://solarscience.msfc.nasa.gov:8004/sparql>
        (BGP
          (triple ?obs rdf:type obs:Observer)
          (triple ?obs person:firstName ?fn)
          (triple ?obs person:lastName ?ln)
        ))
    )
  )))
```

It can be seen that there is a new operator which is not part of official SPARQL: `service`[6]. The implementation of this operator in the query execution engine serializes sub-plans back into SPARQL and uses the protocol to execute it on the endpoint specified by the first parameter. In the next step, the query plan is handed on to the optimizer for which currently only the concepts discussed in Section 6 exist.

5 Sample Queries and Results

For the following sample queries, real-world data of sunspot observations recorded at Kanzelhöhe Solar Observatory (KSO) have been used. The observatory is also a partner in the Austrian Grid project. In the future, the system presented should replace the current archive CESAR (*Central European Solar ARchives*), which is a collaboration between KSO, Hvar Observatory Zagreb, Croatia, and the Astronomical Observatory Trieste, Italy. Because of the flexible architecture other observation sites can easily take part in future. The tests where performed with the following setup: the mediator (and also the test client) where running on a 2.16 GHz Intel Core 2 Duo with 2 GB memory and a 2 MBit link to the remote endpoints. All endpoints where simulated on the same physical host running two AMD Opteron CPUs at 1.6 GHz and 2 GB memory. Local host entries were used to simulate the SPARQL endpoints described in Table 1. The NASA endpoint is imaginary. The table only shows registered data sources which are relevant for the sample queries. The statistics shown were collected by the monitoring service.

The queries are shown in Fig. 2. Query 1 retrieves the first name, the last name, and optionally the e-mail address of scientists who have done observations. Query 2 retrieves all observations ever recorded by Mr. Otruba. This query is used to show a

[6] Special thanks to Andy Seaborne, who implemented this extension after an e-mail discussion about federation of queries in July 2007.

Sunspot observations at KSO	**instances**
endpoint: <http://keas.kso.ac.at:8002/sparql>	
sobs:SunspotRelativeNumbers	9973
sobs:SunExposure	288
sobs:SolarObservationInstrument	7
sobs:Detector	9
obs:Observer	17
Sunspot observations by NASA (imaginary)	**instances**
endpoint: <http://solarscience.msfc.nasa.gov:8004/sparql>	
sobs:SunspotRelativeNumbers	89
obs:Observer	1

Table 1. Endpoints registered when processing samples queries.

distributed join and filter. Query 3 retrieves all sunspot observations recorded in March 1969. Query 4 shows how the mediator's catalog can be accessed. It will list all data sources currently available, the *Virtual Organization*, and the contact person. The plan transformation for Query 2 has been described in the previous section. In Table 2 the response time for the 1st solution, the total execution time (median of 10 samples), and the returned solution mappings are shown. The improvement of the performance is currently on top of the agenda. The results presented in this paper were generated from a prototype not using any optimization of distributed query plans.

Query #	**1st solution**	**total time**	**solutions**
1	162 ms	1.4 s	10
2	290 ms	3.8 s	1,272
3	1,216 ms	37.8 s	43
4	74 ms	0.2 s	6

Table 2. Test results for Query 1–4 and catalog status depicted in Table 1

Because ARQ is using a pipelining concept the response time is very good, even when data has to be retrieved from a remote data source. The reasons why Query 2 and especially Query 3 come off so badly will be discussed in the next section.

6 Optimizations and Future Work

For a mediator, minimizing response time is usually more important than maximizing throughput. Because ARQ is pipelined, response time is very good. However, shipping data is costly, so another goal is the minimization of the amount of data transfered. When using the REST-based SPARQL protocol a second requirement is to minimize the number of required requests. Query 2 and 3 show bad performance mainly because of bad join ordering and not pushing down filters to local sub-plans.

```
SELECT  ?obs ?fname ?lname ?em
WHERE
  { ?obs a obs:Observer ;
         person:lastName   ?lname ;
         person:firstName  ?fname .
    OPTIONAL
      { ?obs  person:email  ?em .}
  }
```

Query 1

```
SELECT  ?dt ?groups ?spots ?r ?fn ?ln
WHERE
  { ?s a sobs:SunspotRelativeNumbers;
         sobs:dateTime     ?dt ;
         sobs:groups       ?groups ;
         sobs:spots        ?spots ;
         sobs:rValue       ?r ;
         obs:byObserver    ?obs .
    ?obs a obs:Observer ;
         person:firstName  ?fn ;
         person:lastName   ?ln .
    FILTER ( ?fn = "Wolfgang" &&
             ?ln = "Otruba" )
  }
```

Query 2

```
SELECT *
WHERE
  { ?s a sobs:SunspotRelativeNumbers;
         sobs:groups      ?groups ;
         sobs:spots       ?spots ;
         sobs:rValue      ?r ;
         obs:description  ?desc;
         obs:byObserver   ?obs ;
         sobs:dateTime    ?dt .
FILTER ( ?dt >=
"1969-03-01T00:00:00"^^xsd:dateTime
      && ?dt <
"1969-04-01T00:00:00"^^xsd:dateTime )
}
```

Query 3

```
SELECT * WHERE {
  ?ds a cat:DataSource .
  ?ds cat:maintainedBy ?vom .
  ?vom a cat:VOMember .
  ?vom person:firstName ?fn .
  ?vom person:lastName ?ln .
  OPTIONAL { ?vom person:email ?e } .
  ?ds cat:providedBy ?vo . }
```

Query 4

Fig. 2. Sample queries

6.1 Optimization of Distributed Query Plans

The following optimization concepts are currently[7] being implemented: push-down of filter expressions, push-down of optional group patterns (becoming left-joins), push-down of local joins whenever possible, and optimization through global join and union re-ordering which is a rather complex task. A holistic approach for finding optimal plans based on *Iterative Dynamic Programming* (IDP) [10] will require heavy modifications to ARQ which should also be discussed in the future. By contrast to implementing static optimization algorithms based on general assumptions, IDP systematically enumerates all possible (equivalent) plans and prunes those with high cost as early as possible during the iteration. At a second stage the implementation of the new service-operator as part of the query execution engine will be extended to support row blocking to reduce the amount of HTTP requests. Some of the algorithms proposed by the database community can be re-used for SPARQL query processing. It is expected that query federation and optimization of distributed SPARQL queries will become more important in future to be able to manage large distributed data stores. Discussions about the *Billion Triples Challenge 2008* [1] indicate a need for scalable base technology which is not limited to local RDF data management.

[7] i.e. at the time of writing this contribution – results are expected to be available for the conference in June 2008 and will be published later on.

In a highly optimized mediator-wrapper system like Garlic [17] or Disco [26], each wrapper provides several operators reflecting local data access and processing capabilities. For instance, an endpoint could support a distributed join operation with another remote node and joins could even be executed in parallel. Exploiting local capabilities would require heavy changes to the current query execution process.

6.2 Future Work

Other future work will be the support for `DESCRIBE`-queries and IRIs as subjects. In future the mediator should also use a OWL-DL reasoner to infer additional types for subject nodes specified in the query pattern. Currently, types have to be explicitly specified for each BGP (more precisely for the first occurrence: the algorithm caches already known types). OWL-DL constraints like for example a qualified cardinality restriction on `obs:byObserver` with `owl:allValuesFrom obs:Observer` would allow the mediator to deduce types of other nodes in the query pattern. Supporting subsumption queries like { `?p a p:Person` } returning all resources that have a sub-type of `p:Person` may considerably inflate the global query plan. Such queries should be supported in future, when global plan optimization has been implemented.

6.3 The Role of Mediators in the Web of Data

As mentioned in the introduction, mediators like SemWIQ can be used to complement Semantic Web search engines and the web of Linked Data [5]. While the traditional hypertext web is separated into the *Surface Web* and the – for search engines hardly reachable – *Deep Web* [13], this separation disappears when browsing through Linked Data. The problem which remains is the fact that the so-called Deep Web is huge. Endpoints in the Web of data may expose large data stores and archives to the public and it will be hard for search engines to index all of these data efficiently. Sindice [27], for instance, is a Semantic Web crawler which is also able to crawl over SPARQL endpoints. However, their authors admitted that this feature is not used at the moment, because of the danger of getting lost in such *black holes*. Fig. 3 shows how SemWIQ can be embedded between a cloud of SPARQL endpoints. The registration component could be extended by a crawler which is autonomously registering new SPARQL endpoints which use RDF Schema or OWL vocabularies to describe their data (e.g. endpoints with D2R). A vocabulary browser which is visualizing all the vocabularies used by registered SPARQL endpoints including freetext search for concepts can be provided to users to examine the virtual data space. Compared to a search engine like Sindice, the mediator allows declarative queries over the complete data space. To maintain its scalibility, it is also possible to use multiple mediators at different locations which may synchronize catalog metadata and statistics over a peer-to-peer overlay network.

7 Conclusion

In this contribution a mediator-based system for virtual data integration based on Semantic Web technology has been presented. The system is primarily developed for sharing scientific data, but because of its generic architecture, it is supposed to be used

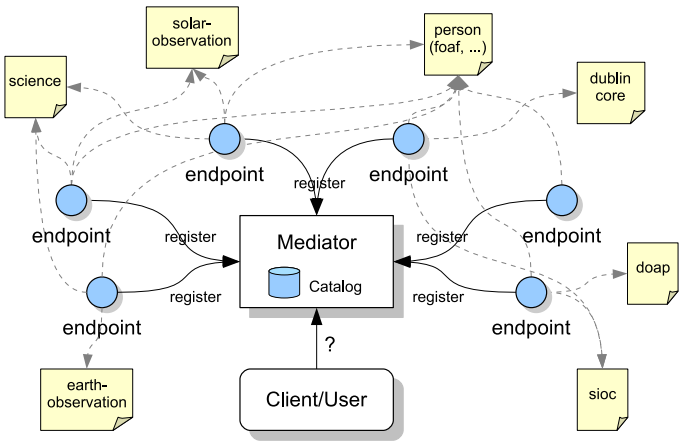

Fig. 3. SemWIQ

for many other Semantic Web applications. In this paper query federation based on SPARQL and Jena/ARQ has been demonstrated in detail and several concepts for query optimization which is currently on the agenda have been discussed. Additional contributions can be expected after the implementation of additional features mentioned before.

Acknowledgements

The work is supported by the *Austrian Grid Project*, funded by the Austrian BMBWK (*Federal Ministry for Education, Science and Culture*), contract GZ 4003/2-VI/4c/2004.

References

1. The Billion Triples Challenge (mailing list archive at Yahoo!). `http://tech.groups.yahoo.com/group/billiontriples/`, 2007. Last visit: Dec 12, 2007.
2. Sören Auer, Chris Bizer, Jens Lehmann, Georgi Kobilarov, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference (ISWC'07)*, volume 4825 of *LNCS*, pages 715–728, Berlin, Heidelberg, November 2007. Springer Verlag.
3. Bastian Quilitz. DARQ – Federated Queries with SPARQL. `http://darq.sourceforge.net/`, 2006. Last visit: Dec 12, 2007.
4. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4):323–364, 1986.
5. T. Berners-Lee, Y. Chen, L. Chilton, and D. Connolly et al. Tabulator: Exploring and analyzing linked data on the semantic web. In *Proceedings of the ISWC Workshop on Semantic Web User Interaction*, 2006.
6. Christian Bizer and Richard Cyganiak. D2RQ – lessons learned. (Position paper for the W3C Workshop on RDF Access to Relational Databases, `http://www.w3.org/2007/03/RdfRDB/papers/d2rq-positionpaper/`), Oct 2007.

7. Chris Bizer and Richard Cyganiak. D2R Server – Publishing Relational Databases on the Semantic Web. In *5th International Semantic Web Conference*, 2006.
8. In Silico Discovery. Semantic discovery system. `http://www.insilicodiscovery.com`, 2007. Last visit: Dec 12, 2007.
9. Donald Kossmann. The State of the Art in Distributed Query Processing. *ACM Comput. Surv.*, 32(4):422–469, 2000.
10. Donald Kossmann and Konrad Stocker. Iterative dynamic programming: a new class of query optimization algorithms. *ACM Trans. Database Syst.*, 25(1):43–82, 2000.
11. I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Lecture Notes in Computer Science*, 2150, 2001.
12. Bernhard Haslhofer. Mediaspaces. `http://www.mediaspaces.info/`, 2007.
13. Bin He, Mitesh Patel, Zhen Zhang, and Kevin Chen-Chuan Chang. Accessing the deep web. *Commun. ACM*, 50(5):94–101, 2007.
14. UK HP Labs, Bristol. Jena – A Semantic Web Framework for Java. `http://jena.sourceforge.net/`. Last visit: March 07.
15. Andreas Langegger, Martin Blöchl, and Wolfram Wöß. Sharing data on the grid using ontologies and distributed SPARQL queries. In *DEXA '07: Proceedings of the 18th International Conference on Database and Expert Systems Applications*, pages 450–454, Washington, DC, USA, 2007. IEEE Computer Society.
16. Andreas Langegger, Wolfram Wöß, and Martin Blöchl. Semantic data access middleware for grids. `http://gsdam.sourceforge.net`. Last visit: Dec 2007.
17. Laura M. Haas, Donald Kossmann, Edward L. Wimmers, and Jun Yang. Optimizing Queries Across Diverse Data Sources. In *Proceedings of the 23th International Conference on Very Large Databases*, pages 276–285, Athens, 1997. VLDB Endowment, Saratoga, Calif.
18. Sergey Melnik. *Generic Model Management: Concepts And Algorithms (Lecture Notes in Computer Science)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2004.
19. A. Miles, T. Baker, and R. Swick. Best practice recipes for publishing RDF vocabularies. `http://www.w3.org/TR/swbp-vocab-pub/`, 2006. Last visit: Dec 12, 2007.
20. N.F. Noy, D.L. Rubin, and M.A. Musen. Making biomedical ontologies and ontology repositories work. *Intelligent Systems*, 19(6):78 – 81, Nov.-Dec. 2004.
21. OpenLink Software. OpenLink Virtuoso. `http://www.openlinksw.com/virtuoso/`. Last visit: March 07.
22. Eric Prud'hommeaux. Optimal RDF access to relational databases. `http://www.w3.org/2004/04/30-RDF-RDB-access/`, April 2004.
23. Eric Prud'hommeaux. Federated SPARQL. `http://www.w3.org/2007/05/SPARQLfed/`, May 2007.
24. Kai-Uwe Sattler, Ingolf Geist, and Eike Schallehn. Concept-based querying in mediator systems. *The VLDB Journal*, 14(1):97–111, 2005.
25. He Tan and Patrick Lambrix. A method for recommending ontology alignment strategies. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea*, volume 4825 of *LNCS*, pages 491–504, Berlin, Heidelberg, 2007. Springer.
26. A. Tomasic, L. Raschid, and P. Valduriez. Scaling heterogeneous databases and the design of disco. *ICDCS*, 00:449, 1996.
27. Giovanni Tummarello, Renaud Delbru, and Eyal Oren. Sindice.com: Weaving the open linked data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, November 2007.
28. W3C. SPARQL Query Language for RDF, W3C Proposed Recommendation. `http://www.w3.org/TR/rdf-sparql-query/`. Last visit: May 2007.
29. Gio Wiederhold. Mediators in the architecture of future information systems. *Computer*, 25(3):38–49, 1992.