Robust service-based semantic querying to distributed heterogeneous databases

Carlos Buil-Aranda*, Oscar Corcho* and Amy Krause**
*Ontology Engineering Group, Departamento de Inteligencia Artificial Universidad Politécnica de Madrid, **EPCC, The University of Edinburgh
{cbuil, ocorcho}@fi.upm.es, a.krause@epcc.ed.ac.uk

Abstract

The amount of semantic data on the Web has increased exponentially in the last years. One of the main reasons for this is the use of RDB2RDF systems, which generate RDF data from relational databases. Most of the work on these systems has focused on increasing the expressivity of query languages, analyzing their coverage over databases and generating sound, complete and efficient query plans. However, there are still important problems associated to them, especially in terms of robustness and in the handling of data distribution. In this paper we describe how we have integrated one RDB2RDF system (ODEMapster) with OGSA-DAI in order to overcome these problems.

1. Introduction

In the last years there has been an important increased of RDF data publicly and privately available, due to initiatives like Linked Data [6] and to the availability of RDB2RDF systems [5], which allow accessing relational data by using query languages like SPARQL. These systems map relational databases to ontologies, providing semantic access to the content stored in a relational model.

One important problem with this type of systems, which is now gaining relevance, is their robustness. RDB2RDF systems normally transform the SPARQL query into a set of SQL queries that are then sent to the relational database management system, and then combine the obtained result into the SPARQL result format. When these systems are used to extract instances from a small database, or from a large database but with a small amount of instances, the extraction process is usually quite fast. However, when the amount of data to be transformed and delivered (as RDF or in the SPARQL result format) is large, the time needed to obtain the data and the possibility of having errors in the connection to the system are also high.

The distribution of data resources is also a common characteristic in applications that use this type of systems, with data sources that may belong to different organizations, be placed in different geographic locations, and with different authorization requirements. Typically the management of all these distributed data sources is tedious: clients have to manage connections, authenticate themselves, deal with specific security policies, etc. All these issues are normally implemented ad-hoc at the system client side.

In this paper we describe a robust service-based semantic querying system to distributed data resources, based on two widely used technologies: RDB2RDF systems, and WS-DAI [3], a Web service specification that allows users to access distributed data resources.

This paper is organized as follows: section 2 describes the functionalities of RDB2RDF tools. A short description of one implementation of one of these systems is presented (R2O and ODEMapster). Section 3 presents a brief description of the WS-DAI specification, and of one of its implementations (OGSA-DAI). In section 4 we present how the combination of these two systems is done. In section 5 we explain how the integration of heterogeneous data sources can be done by means of data processing workflows. In section 6 we show a brief experimentation that we did with our system, and in section 7 we present the conclusions and the future plans for our system.

2. RDB2RDF systems

RDB2RDF systems [5] are systems that map relational databases to RDF, providing semantic access to existing relational databases by mapping relational entities to RDF classes and properties. There are two features that allow characterizing these systems:

- Mapping creation: mappings can be created automatically, normally resembling the initial database structure, or manually, normally driven by a set of domain ontologies.

- Materialization vs. virtualization of databases: these systems can normally operate in two different modes: materializing (dumping) the database into RDF, and virtualizing the database, by using RDF query languages directly without creating any instance in the ontology.

Figure 1 depicts these two working modes. In both cases, the query processor interprets the mappings and accesses the databases according to them.

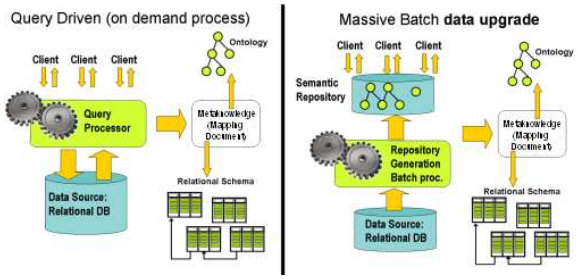

Figure 1: RDB2RDF systems working modes

There are many systems that map relational databases to RDF (R2O [1], D2R [2], etc.). We will describe one of them, R2O, and its associated processor. A more exhaustive list can be found in [5].

2.1 R2O and ODEMapster

R2O [1] is a declarative, XML-based language that allows specifying mappings between the elements of an ontology (concepts, attributes and relations) and the elements of a relational database (relations and attributes). The R2O language is DBMS independent and allows the mapping files to be used in different databases. The main elements of R2O are conditions and operations and a rule-style mapping definition for attributes.

Conditions allow the mapping between relational tuples or database records to instances of the target ontology and operations describe the transformations needed to create ontology instances from database instances. Attribute mappings are defined as sets of if-then rules. These rules allow the conditional generation of attribute values as well as multivaluation. The structure of an attribute mapping definition is described by the following example. The value of the ontology attribute type is calculated based on the application of the set of rules (selector): If the condition part (applies-if) is verified, then the action part (after transform) is executed to generate a value.

ODEMapster is the engine in charge of transforming semantic queries into SQL queries according to the mappings specified in the R2O language. This process can be done in materialized and virtualized ways.

3. Service oriented data access (WS-DAI)

The **WS-DAI** specification defines interfaces to access data resources as web services, and is a recommendation of the Open Grid Forum. The general WS-DAI specification has two extended realizations, one for accessing relational databases (WS-DAIR) and another for accessing XML databases (WS-DAIX). And work is being done in providing another extended realization for RDF data (WS-DAI-RDF).

The key elements of the WS-DAI specification are data services. A data service is a Web service that implements one or more of the DAIS-WG specified interfaces to provide access to data resources. A data resource is a source of data (relational databases, XML databases, file systems). The WS-DAI specification defines concepts, properties and messages that can be used in the definition of data services. It does not provide elements to manage these data sources (means for managing DBMS), only for accessing them. Data resources are divided in externally managed data resources (the data is stored using an existing data management system) and service managed data resource (it does not normally exist outside the service-oriented middleware and its lifetime is managed in ways that are specified in the DAIS specifications).

Data resource access modes: There are two different ways for accessing data resources. The first one is direct access to data resources with which the user accesses these resources like a regular service. The user sends a request to the resource with a SQL query in the parameters. The result of the web service is a row set with data requested.

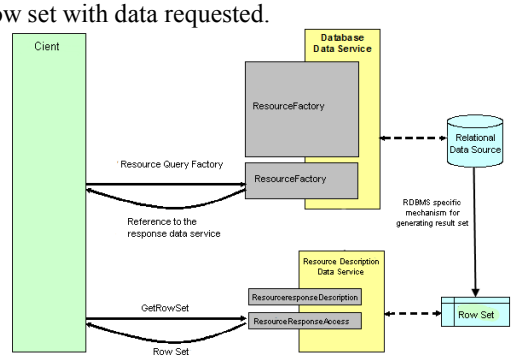

Figure 2: Indirect access to data resources

The second way for accessing data resources is the indirect access. Indirect access implements a factory pattern for the user data requests. A user queries a data resource and the data resource creates a new data resource with the results of this query. This new resource will be accessible for the client. Figure 2 represents this situation. The WS-DAI specification does not define any operation for implementing the indirect access but defines the template to be followed.

OGSA-DAI [4] is an extensible framework for accessing distributed and heterogeneous data. It was primarily intended as the WS-DAI reference implementation but it has evolved differently, extending it. It executes data-centric workflows involving heterogeneous data resources for the purposes of data access, integration, transformation and delivery within a Grid. It is intended as a toolkit for building higher-level application-specific data services. The user is able to query different types of data sources such as distributed relational databases, XML databases or file systems and process the results

OGSA-DAI relies in two key elements. The first element is data resources, implementing some of the WS-DAI methods. These resources can represent relational databases, XML databases or file systems. The data sources are distributed and for making them available it is necessary to create an OGSA-DAI resource which connects the OGSA-DAI platform to the data source.

The second key element of OGSA-DAI is activities which are operations or named units of functionalities (data goes in, something is done, data comes out). The user for querying the OGSA-DAI resources has to create activities Finally users can combine activities to create workflows. The workflows can be created by combining inputs and outputs from activities that access the different resources within a Grid.

OGSA-DAI is integrated in Apache Tomcat and within the Globus toolkit and is used in OMII-UK, the UK e-Science platform.

4. Service-based data access to improve the robustness of RDB2RDF systems

In this section we describe how we combine the technologies described in section 2 and 3 (ODEMapster and OGSA-DAI) for providing robust and distributed semantic access to relational data.

4.1 Extending OGSA-DAI data resources: the R2O resource

The first step to use an RDB2RDF system (e.g., R2O) in this context is to extend OGSA-DAI with a new data resource. This data resource is the ODEMapster engine, which processes the R2O mapping files. The ODEMapster resource will configure and access the relational database through a mapped ontology using the R2O language. Figure 2 represents how OGSA-DAI Web services are extended with the new resource type.

As described in section 2, the query process can be done following a materialized or virtualization mode. These two modes only make sense in a desktop application environment but in this document we focus in the distribution of resources. These resources require an adaptation of the querying methods explained in the next lines.

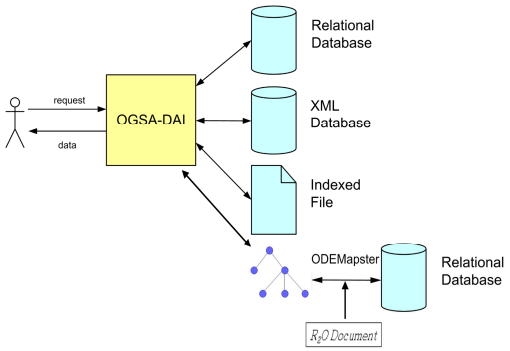

Figure 3: OGSA-DAI with the new resource

4.1.1 OGSA-DAI R2O extension

The new resource has been created in a similar way to the JDBCResource already existing in OGSA-DAI. This resource provides users with a way to query a relational database and this is one of the purposes of the R2O resource: to provide a way for querying a relational database but using a RDF query language and allowing a batch processing mode.

4.1.2 R2O resource

In order to create the new R2O resource we have to implement several OGSA-DAI's interfaces, such as uk.org.ogsadai.resource.dataresource.DataResource, which contains the main methods that a data resource must have. The classes developed for providing distributed access to ODEMapster are:

R2OResource: this class implements the OGSA-DAI DataResource interface. The implemented methods are *getState()*, which returns the configuration parameters of the "R2OResource", including the necessary elements for the correct execution of the data resource; *initialize()*, which creates the R2O data resource and initializes it allowing the OGSA-DAI persistence and configuration components to configure the data resource class with its configuration (the R2O mapping file and other configuration files); and *getR2OResult()*, in charge of executing the ODEMapster engine with the previously specified parameters. Other methods for managing the lifecycle of the resource are also implemented in this class.

R2OConnectionProvider: this class gives access to the R2O resource that is necessary to create an OGSA-DAI resource accessor. When an activity wants to access to the data resource, OGSA-DAI is in charge of going to the resource and provides this activity with the

methods *getConnection()* and *releaseConnection()*. R2OConnectionProvider is the interface that must be implemented and that provides these methods.

R2OResourceState: this class represents the state of the R2O data resource. On data resource creation the associated DataResourceState (in this case the R2O resource) object is registered with OGSA-DAI persistence and configuration components. Any changes to this object are persisted. Hence if the configuration of a data resource is to be persisted it should be reflected in the associated DataResourceState.

MapsterConnector: this class is in charge of processing the R2O mapping file. It runs the ODEMapster engine and starts creating the mapping in the ontology.

4.1.3 R2O Activities

The previously defined classes are the classes that implement the new R2O data resource. But in order to use this resource it is necessary to create both server and client side OGSA-DAI activities.

R2O server activity: this activity is in charge of connecting the client side activity and the R2O data resource. Not only does it connect the client to the data resource but also it creates activities that manipulate the outputs from the resource to provide the end user with a more elaborated response. In our case the activity created only starts the engine that creates the instances in the ontology from the database. The activity extends the *ActivityBase* class, which is responsible for providing access to input and output pipes, activity contracts and simple validation functionality. For interacting with the R2O resource the activity has to implement the ResourceActivity class and has to contact with the OGSA-DAI resource accessor.

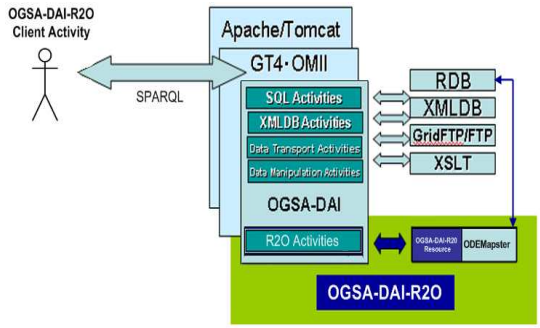

Figure 4: OGSA-DAI-R2O extension (based on OGSA-DAI-RDF extension figures).

R2O client activity: The client side activity connects to the server side activity, requesting the execution of the R2O processor. The activity gets as response a pointer where the results are being stored. Once the results are available it is possible to compose the output of the activity with other outputs, create workflows, etc. Figure 4 shows the interactions of the activities (client and server side activities) with the R2O resource and its location within OGSA-DAI and the possible servers.

4.1.4 How it works

The user starts its application invoking OGSA-DAI by sending a DataRequestExecutionResource. This is the resource in charge of calling resources that need to be executed. In our case the "R2OResource" will be invoked. Now the server receives the request from the client and invokes the R2OActivity at the server side. This activity communicates with the R2OResource invoking the *getR2OResult()* method making ODEMapster to run the R2O mappings and generate the RDF instances from the relational database. This activity is also prepared to manage connections to the databases and be extended with more functionalities (which are detailed in the future work section).

5. Creating workflows with OGSA-DAI-R2O: managing distribution

In OGSA-DAI there are three types of workflows. The first type is the pipeline workflow type, with which a set of chained activities are executed in parallel with data flowing between the activities. Other workflow types are sequence workflows, where a set of sub workflows are executed in sequence, and parallel workflows, which contain a set of workflows executed in parallel. These types of workflow can be also applied to the new implementation that we have created and that was described in the previous section, allowing semantic queries across several databases. Query processing activities are done at the server side and when the processing of the query is done the client has two ways of accessing the results. If the invocation was synchronous a stream of data with the result is returned. If the invocation was asynchronous a notification is returned to the user. It is possible to query the state of the current state of the execution using the OGSA-DAI's request execution resource (which was previously configured).

OGSA-DAI provides lifetime operations for managing data resources. When an asynchronous query to a resource is requested, the result can be sent to the same query request, to an FTP or to a data source resource for retrieval by a client or another OGSA-DAI server, and can be retrieved immediately or asynchronously. Clients can, using the new data source,

access to it using small data sets or manage their lifetime with the methods provided by OGSA-DAI.

6. Experimentation

Several experiments have been performed with the new R2O resource for OGSA-DAI to check the feasibility of our implementation. These experiments have not been exhaustive due to the fact that the development is still in an early stage, and for instance we have not made large experiments with semantic data workflows. The experiments consisted in accessing the Digitalis database[1], which contains freely available information about pharmaceutical products in Spain. It contains 22 tables and about 46000 registries. The time for creating a complete materialized view over this database with ODEMapster is around two minutes.

Our experiment consisted in executing the OGSA-DAI-R2O resource with this database. Obviously using the indirect access to R2O we needed around 2 seconds to receive the response from the resource, and around two minutes to get access to the new data resource containing the instances. The resource in the synchronous call was returned as a stream of data meanwhile in the asynchronous execution it was returned a new OGSA-DAI source of data.

7. Conclusions

In this article we have shown how we created a system for accessing distributed relational databases by using an RDB2RDF system. The WS-DAI specification describes how to access distributed data sources using Web services. We used OGSA-DAI, which implements most of the interfaces defined at the WS-DAI specification to create a new data resource at the OGSA-DAI server side. This new OGSA-DAI data resource accesses an RDB2RDF engine (ODEMapster). This new resource is in charge of accessing a relational database using SPARQL, initially in batch mode. Using the WS-DAI recommendation of indirect access to data resources, the end user submits a query and can do other processing while at the server this query is performed. This system provides robust access allowing users to manage the lifetime of the data resources and their results, by means of the lifetime methods provided by OGSA-DAI.

This development is still in an early stage. Technically some improvements need to be done, among others: provide to the users the ability of manage database connections at the client side, provide direct query activities, and most important, align the outputs from the OGSA-DAI-R2O activities with the other OGSA-DAI activities. This alignment would allow users to create integration workflows with different data sources like RDF data, XML data and relational data.

The development of the OGSA-DAI-R2O resource will continue for providing the integration functionalities described before.

[1] http://www.msc.es/profesionales/farmacia/nomenclatorDI.htm

8. ACKNOWLEDGMENTS

Help for this work was provided by the OGSA-DAI team. This work is supported by the ADMIRE project (FP7-ICT-215024).

9. REFERENCES

[1] Barrasa J, Gómez-Pérez A. Upgrading relational legacy data to the semantic web. In WWW2006, pp. 1069-1070, Edinburgh, United Kingdom, May 2006.

[2] Bizer C, Cyganiak R. D2RQ - Lessons Learned. Position paper for the W3C Workshop on RDF Access to Relational Databases, Cambridge, USA, October 2007.

[3] Antonioletti M, Krause A, Paton NW, Eisenberg A, Laws S, Malaika S, Melton J, Pearson D. The WS-DAI family of specifications for web service data access and integration, ACM SIGMOD Record, v.35 n.1, March 2006

[4] Antonioletti, M., Chue Hong, N.P., Hume, A.C., Jackson, M., Karasavvas, K., Krause, A., Schopf, J.M., Atkinson, M.P., Dobrzelecki, B., Illingworth, M., McDonnell, N., Parsons, M. and Theocharopoulos, E.. OGSA-DAI 3.0 - The Whats and the Whys, Proceedings of the UK e-Science All Hands Meeting 2007, pp. 158-165, September 2007.

[5] Satya S. Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Sören Auer, Juan Sequeda, Ahmed Ezzat. A Survey of Current Approaches for Mapping of Relational Databases to RDF. 31-01-2009. W3C RDB2RDF Incubator Group.

[6] Bizer C, Heath T, Idehen K, Berners-Lee T. Linked data on the web (LDOW2008). In WWW 2008 pp. 1265-1266, Beijing, China.