DATA INTEGRATION AND QUERY REFORMULATION IN SERVICE-BASED GRIDS

Carmela Comito and Domenico Talia
DEIS, University of Calabria, Italy
ccomito@deis.unical.it
talia@deis.unical.it

Anastasios Gounaris and Rizos Sakellariou
School of Computer Science, University of Manchester, UK
gounaris@cs.man.ac.uk
rizos@cs.man.ac.uk

Abstract

This paper describes the XMAP data integration framework and query reformulation algorithm, provides insights into the performance of the algorithm, and about its use in implementing query processing services. Here we propose an approach for data integration-enabled distributed query processing on Grids by embedding the XMAP reformulation algorithm within the OGSA-DQP distributed query processor. To this aim we exploit the OGSA-DQP XML representation of relational schemas by applying the XMAP algorithm on them. Moreover, we introduce a technique to rewrite an XPath query into an equivalent OQL one. Finally, the paper presents a roadmap for the integration system implementation aiming at constructing an extended set of services that will allow users to submit queries over a single database and receive the results from multiple databases that are semantically correlated with the former one.

Keywords: XML databases, semantic data integration, schema mappings, distributed query processing, Grid services.

1. Introduction

The Grid offers new opportunities and raises new challenges in data management that originate from the large scale, dynamic, autonomous, and distributed nature of data sources. A Grid can include related data resources maintained in different syntaxes, managed by different software systems, and accessible through different protocols and interfaces. Due to this diversity in data resources, one of the most demanding issues in managing data on Grids is reconciliation of data heterogeneity [11]. Therefore, in order to provide facilities for addressing requests over multiple heterogeneous data sources, it is necessary to provide data integration models and mechanisms.

Data integration is the flexible and managed federation, analysis, and processing of data from different distributed sources. In particular, the increase in availability of web-based data sources has led to new challenges in data integration systems for obtaining decentralized, wide-scale sharing of data, preserving semantics. These new needs in data integration systems are also felt in Grid settings. In a Grid, a centralized structure for coordinating all the nodes is not efficient because it can represent a bottleneck and, more importantly, it cannot accommodate the dynamic and distributed nature of Grid resources.

The Grid community is devoting great attention toward the management of structured and semi-structured data such as relational and XML data. Two significant examples of such efforts are the *OGSA Data Access and Integration* (OGSA-DAI) [3] and the *OGSA Distributed Query Processor* (OGSA-DQP) [2] projects. However, till today only few projects (e.g., [8, 6]) actually meet schema-integration requirements necessary for establishing semantic connections among heterogeneous data sources.

For these reasons, we propose the use of the *XMAP* framework [9] for integrating heterogeneous data sources distributed over a Grid. By means of this framework, we aim at developing a decentralized network of semantically related schemas that enables the formulation of distributed queries over heterogeneous data sources. We designed a method to combine and query XML documents through a decentralized point-to-point mediation process among the different data sources based on schema mappings. We offer a decentralized service-based architecture that exposes this XML integration formalism as an e-Service. The infrastructure proposed exploits the middleware provided by OGSA-DQP and OGSA-DAI, building on top of them schema-integration services.

The remainder of the paper is organized as follows. Section 2 presents a short analysis of data integration systems focusing on specific issues related to Grids. Section 3 presents the XMAP integration framework; the underlying integration model and the XMAP query reformulation algorithm are described. The OGSA-DQP and OGSA-DAI existing query processing services are outlined in Section

4. Section 5 presents an example of applying the XMAP algorithm to OGSA-DQP, whereas Section 6 introduces the approach proposed to rewrite an XPath query into an equivalent OQL one. Finally, Section 8 concludes the paper.

2. Data Integration in Grids

The goal of a data integration system is to combine heterogeneous data residing at different sites by providing a unified view of this data. The two main approaches to data integration are federated database management systems (FDBMSs) and traditional mediator/wrapper-based integration systems.

A federated database management system (FDBMS) [19] is a collection of cooperating but autonomous component database systems (DBSs). The DBMS of a component DBS, or component DBMS, can be a centralized or distributed DBMS or another FDBMS. The component DBMSs can differ in different aspects such as data models, query languages, and transaction management capabilities.

Traditional data integration systems [17] are characterized by an architecture based on one or more mediated schemas and a set of sources. Each source contains data, while every mediated schema provides a reconciled, integrated, and virtual view of the underlying sources. Moreover, the system includes a set of source descriptions that provide semantic mappings between the relations in the source schemas and the relations in the mediated schemas [18] .

Data integration on Grids presents a twofold characterization:

1 data integration is a key issue for exploiting the availability of large, heterogeneous, distributed and highly dynamic data volumes on Grids;

2 integration formalisms can benefit from an OGSA-based Grid infrastructure, since it facilitates dynamic discovery, allocation, access, and use of both data sources and computational resources, as required to support computationally demanding database operations such as query reformulation, compilation and evaluation.

Data integration on Grids has to deal with unpredictable, highly dynamic data volumes provided by unpredictable membership of nodes that happen to be participating at any given time. So, traditional approaches to data integration, such as FDBMS [19] and the use of mediator/wrapper middleware [18] , are not suitable in Grid settings.

The federation approach is a rather rigid configuration where resources allocation is static and optimization cannot take advantage of evolving circumstances in the execution environment. The design of mediator/wrapper integration systems must be done globally and the coordination of mediators has been done by a central administrator which is an obstacle to the exploitation of evolving characteristics of dynamic environments. As a consequence, data sources cannot change often and significantly, otherwise they might violate the mappings to the mediated schema.

The rise in availability of web-based data sources has led to new challenges in data integration systems in order to obtain decentralized, wide-scale sharing of semantically-related data. Recently, several works on data management in peer-to-peer (P2P) systems are pursuing this approach [4, 7, 13, 14, 15]. All these systems focus on an integration approach that excludes a global schema: each peer represents an autonomous information system, and data integration is achieved by establishing mappings among the various peers.

To the best of our knowledge, there are only few works designed to provide schema-integration in Grids. The most notable ones are *Hyper* [8] and *GDMS* [6] . Both systems are based on the same approach that we have used ourselves: building data integration services by extending the reference implementation of OGSA-DAI. However, the *Grid Data Mediation Service* (GDMS) uses a wrapper/mediator approach based on a global schema. GDMS presents heterogeneous, distributed data sources as one logical virtual data source in the form of an OGSA-DAI service. For its part, *Hyper* is a framework that integrates relational data in P2P systems built on Grid infrastructures. As in other P2P integration systems, the integration is achieved without using any hierarchical structure for establishing mappings among the autonomous peers. That framework uses a simple relational language for expressing both the schemas and the mappings. By comparison, our integration model follows, like Hyper, an approach not based on a hierarchical structure. However, differently from Hyper, it focuses on XML data sources and is based on schema-mappings that associate paths in different schemas.

3. XMAP: A Decentralized XML Data Integration Framework

The primary design goal the XMAP framework is to develop a decentralized network of semantically related schemas that enables the formulation of queries over heterogeneous, distributed data sources. The environment is modeled as a system composed of a number of Grid nodes, where each node can hold one or more XML databases. These nodes are connected to each other through declarative mappings rules.

The XMAP integration [9] model is based on schema mappings to translate queries between different schemas. The goal of a schema mapping is to capture structural as well as terminological correspondences between schemas. Thus, in [9] , we propose a decentralized approach inspired by [14] where the mapping rules are established directly among source schemas without relying on a central mediator or a hierarchy of mediators. The specification of mappings is thus flexible and scalable: each source schema is directly connected to only a small number of other schemas. However, it remains reachable from all other schemas that belong to its transitive closure. In other words, the system supports two different kinds of mapping to connect schemas semantically: point-to-point mappings and transitive mappings. In transitive mappings, data sources are related through one or more "mediator schemas".

We address structural heterogeneity among XML data sources by associating paths in different schemas. Mappings are specified as path expressions that relate a specific element or attribute (together with its path) in the source schema to related elements or attributes in the destination schema.. The mapping rules are specified in XML documents called XMAP documents. Each source schema in the framework is associated to an XMAP document containing all the mapping rules related to it.

The key issue of the XMAP framework is the XPath reformulation algorithm: when a query is posed over the schema of a node, the system will utilize data from any node that is transitively connected by semantic mappings, by chaining mappings, and reformulate the given query expanding and translating it into appropriate queries over semantically related nodes. Every time the reformulation reaches a node that stores no redundant data, the appropriate query is posed on that node, and additional answers may be found. As a first step, we consider only a subset of the full XPath language.

We have implemented the XMAP reformulation algorithm in Java and evaluated its performance by executing a set of experiments. Our goals with these experiments are to demonstrate the feasibility of the XMAP integration model and to identify the key elements determining the behavior of the algorithm. The experiments discussed here have been performed to evaluate the execution time of the reformulation algorithm on the basis of some parameters like the *rank* of the semantic network, the *mapping topology*, and the *input query*. The rank corresponds to the average rank of a node in the network, i.e., the average number of mappings per node. A higher rank corresponds to a more interconnected network. The topology of the mappings is the way how mappings are established among the different nodes, it is the shape of the semantic network.

The experimental results were obtained by averaging the output of 1000 runs of a given configuration. Due to lacks of space here we report only few results of the performed evaluations .

Figure 1 shows the total reformulation time as function of the number of paths in the query for three different ranks. The main result showed in the figure is the low time needed to execute the algorithm that ranges from few milliseconds when a single path is involved to one second where a larger number of paths are to be considered. As should be noted from that figure, for a given rank value, the running times are lower when the mappings guarantee a uniform semantic connection This happens because some mappings provide better connectivity than others.

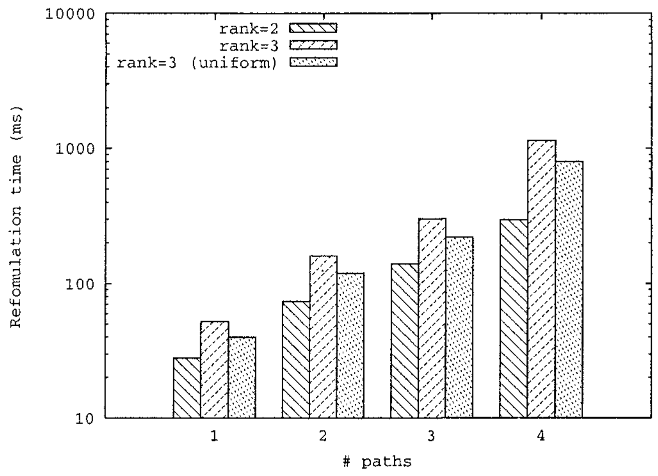

Figure 1. Total reformulation time as function of the number of paths in the query for three different ranks.

In another set of experiments in which we have used the mapping topology as a free variable (see Figure 2), we deduced that for large-scale, highly dynamic networks the best solution is to organize mappings in random topologies with a low average rank. A random topology produces smaller reformulation steps (that is, a smaller number of recursive invocations of the algorithms) that results in lower reformulation times so guaranteeing scalability, fault-tolerance, and flexibility.

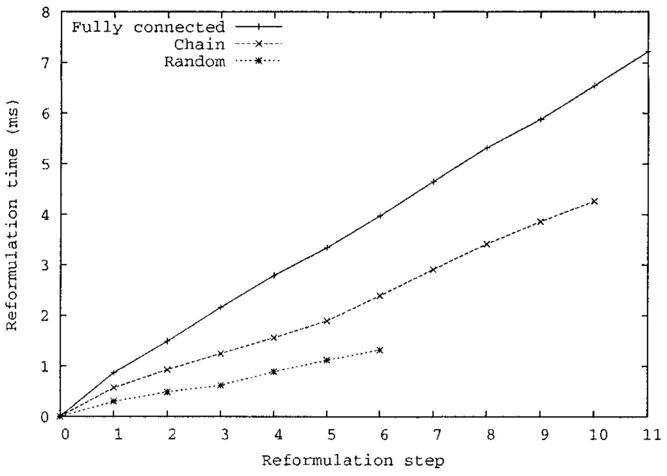

Figure 2. Time to first reformulation for the different topologies.

4. Introduction to Grid query processing services

The Grid community is devoting great attention toward the management of structured and semi-structured data such as relational and XML data. Two significant examples of such efforts are the *OGSA Data Access and Integration* (OGSA-DAI) [3] and the *OGSA Distributed Query Processor* (OGSA-DQP) projects [2].

OGSA-DAI provides uniform service interfaces for data access and integration via the Grid. Through the OGSA-DAI interfaces disparate, heterogeneous data resources can be accessed and controlled as though they were a single logical resource. OGSA-DAI components also offer the potential to be used as basic primitives in the creation of sophisticated higher-level services that offer the capabilities of data federation and distributed query processing within a Virtual Organization (VO).

OGSA-DAI can be considered logically as a number of co-operating Grid services. These Grid services act as proxies for the systems that actually hold the data that is relational databases (for example MySQL) and XML databases (for example Xindice). Clients requiring data held within such databases access the data via the OGSA-DAI Grid services. The Grid Data Service (GDS) is the primary OGSA-DAI service. GDSs provide access to data resources using a document-oriented model: a client submits a data retrieval or update request in the form of an XML document, the GDS executes the request and returns an XML document holding the results of the request.

OGSA-DQP is an open source service-based Distributed Query Processor that supports the evaluation of queries over collections of potentially remote data access and analysis services. Here query compilation, optimisation and evaluation are viewed (and implemented) as invocations of OGSA-compliant GSs. OGSA-DQP supports the evaluation of queries expressed in a declarative language over one or more existing services. These services are likely to include mainly database services, but may also include other computational services. As such, OGSA-DQP supports service orchestration and can be seen as complementary to other infrastructures for service orchestration, such as workflow languages.

OGSA-DQP uses Grid Data Services (GDSs) provided by OGSA-DAI to hide data source heterogeneities and ensure consistent access to data and metadata. Notably, it also adapts techniques from parallel databases to provide implicit parallelism for complex data-intensive requests. The current version of OGSA-DQP, OGSA-DQP 3.0, uses Globus Toolkit 4.0 for grid service creation and management. Thus OGSA-DQP builds upon an OGSA-DAI distribution that is based on the WSRF infrastructure. In addition, both GT4.0 and OGSA-

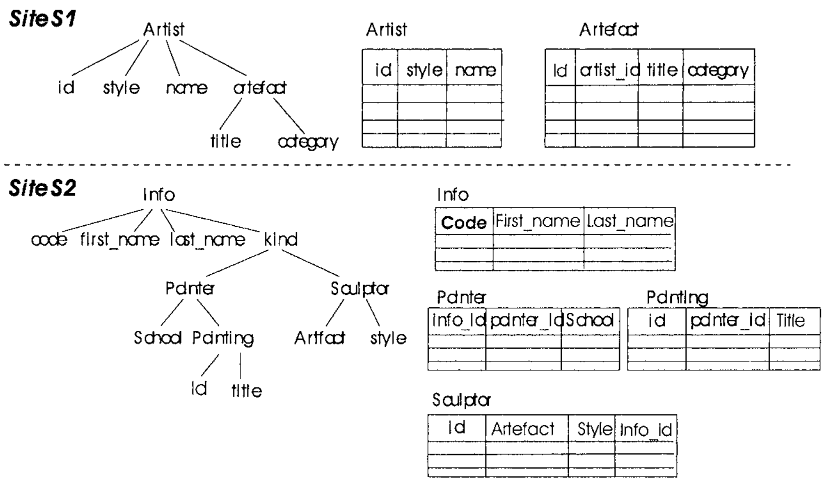
SiteS1
Artist
id style name artefact
title category
Artist
id style name
Artefact
Id artist_id title category
SiteS2
Info
code first_name last_name kind
Painter Sculptor
School Painting Artfact style
Id title
Info
Code First_name Last_name
Painter
Info_Id painter_Id School
Painting
id painter_id Title
Sculptor
Id Artefact Style Info_id

Figure 3. The example schemas.

DAI require a web service container (e.g. Axis) and a web server (such as Apache Tomcat) below them.

OGSA-DQP provides two additional types of services, Grid Distributed Query Services (GDQSs) and Grid Query Evaluation Services (GQESs). The former are visible to end users through a GUI client, accept queries from them, construct and optimise the corresponding query plans and coordinate the query execution. GQESs implement the query engine, interact with other services (such as GDSs, ordinary Web Services and other instances of GQESs), and are responsible for the execution of the query plans created by GDQSs.

5. Integrating the XMAP algorithm in service-based Grids: A walk-through example

The XMAP algorithm can be used for data integration-enabled query processing in OGSA-DQP. This example aims to show how the XMAP algorithm can be applied on top of the OGSA-DAI and OGSA-DQP services. In the example, we will assume that the underlying databases, of which the XML representation of the schema is processed by the XMAP algorithm, are, in fact, relational databases, like those supported by the current version of OGSA-DQP.

We assume that there are two sites, each holding a separate, autonomous database that contains information about artists and their works. Figure 3 presents two self-explanatory views: one hierarchical (for native XML databases), and one tabular (for object-relational DBMSs).

In OGSA-DQP, the table schemas are retrieved and exposed in the form of XML documents, as shown in Figure 4.

```
<databaseSchema dbname="S1">
    <table name="Artist">
        <column name="id" />
        <column name="style" />
        <column name="name" />
        <primaryKey>
            <columnName>id</columnName>
        </primaryKey>
    </table>
    <table name="Artefact">
        <column name="artist_id" />
        <column name="title" />
        <column name="category" />
    </table>
</databaseSchema>

<databaseSchema dbname="S2">
    <table name="Info">
        <column name="id" />
        <column name="code" />
        <column name="first_name" />
        <column name="last_name" />
        <column name="kind" />
        <primaryKey>
            <columnName>id</columnName>
        </primaryKey>
    </table>
    <table name="Painter">
        <column name="painter_id" />
        <column name="info_id" />
        <column name="school" />
        <primaryKey>
            <columnName>painter_id</columnName>
        </primaryKey>
    </table>
    <table name="Painting">
        <column name="painter_id" />
        <column name="title" />
        <primaryKey>
            <columnName>title</columnName>
        </primaryKey>
    </table>
    <table name="Sculptor">
        <column name="info_id" />
        <column name="artefact" />
        <column name="style" />
    </table>
</databaseSchema>
```

Figure 4. The XML representation of the schemas of the example databases.

The XMAP mappings need to capture the semantic relationships between the data fields in different databases, including the primary and foreign keys. This can be done in two ways, which are illustrated in Figures 5 and 6, respectively. Both the ways seem to be feasible. However, the second one is slightly more comprehensible, and thus more desirable.

The actual query reformulation occurs exactly as described in [9] . Initially, users submit XPath queries that refer to a single physical database. E.g., the query `/S1/Artist [style="Cubism"]/name` extracts the names of the artists whose style is *Cubism* and their data is stored in the *S1* database. Similarly, the query `/S1/Artefact/title` returns the titles of the artifacts in the same database. When the XMAP algorithm is applied for the second query, two more XPath expressions will be created that refer to the *S2* database:

```
i)
databaseSchema[@dbname=S1]/table[@name=Artist]/column[@name=style]
->
     databaseSchema[@dbname=S2]/table[@name=Painter]/column[@name=school],
     databaseSchema[@dbname=S2]/table[@name=Sculptor]/column[@name=style]
ii)
databaseSchema[@dbname=S1]/table[@name=Artefact]/column[@name=title]
->
     databaseSchema[@dbname=S2]/table[@name=Painting]/column[@name=title],
     databaseSchema[@dbname=S2]/table[@name=Sculptor]/column[@name=artefact]

iii) databaseSchema[@dbname=S1]/table[@name=Artist/column[@name=id
->
     databaseSchema[@dbname=S2]/table[@name=Info/column[@name=id]

iv)
databaseSchema[@dbname=S1]/table[@name=Artefact]/column[@name=artist_id]
->
     databaseSchema[@dbname=S2]/table[@name=Painter]/column[@name=info_id],
     databaseSchema[@dbname=S2]/table[@name=Sculptor]/column[@name=info_id]
```

Figure 5. The XMAP mappings.

```
i) S1/Artist/style -> S2/Painter/school, S2/Sculptor/style

ii)S1/Artefact/title -> S2/Painting/title, S2/Sculptor/artefact

iii) S1/Artist/id -> S2/Info/id

iv) S1/Artefact/artist_id->S2/Painter/info_id,S2/Sculptor/info_id
```

Figure 6. A simpler form of the XMAP mappings.

`/S2/Painting/Title` and `/S2/Sculptor/Artefact`. At the back-end, the following queries will be submitted to the underlying databases (in SQL-like format):

```
select title from Artefact;
select title from Painting; and
select Artefact from Sculptor;
```

Note that the mapping of simple XPath expressions to SQL/OQL is feasible [16].

6. XPath to OQL mapping

OGSA-DQP through the *GDQS* service should be capable of accepting XPath queries, and of transforming these XPath queries to OQL before parsing, compiling, optimising and scheduling them. Such a transformation falls in an active research area (e.g., [12, 5]), and is implemented as an additional component within the query compiler. In general, the set of meaningful XPath queries over the XML representation of the schema of relational databases supported by OGSA-DQP fits into the following template:

$$/database\_A[predicate\_A]/table\_A[predicate\_B]/column\_A$$

where

$$predicate\_A ::= table\_pred\_A[column\_pred\_A = value\_pred\_A],\ and$$

$$predicate\_B ::= column\_pred\_B = value\_pred\_B$$

As such, the mapping to the `select`, `from`, `where` clauses of OQL is straightforward. *column_A* defines the `select` attribute, whereas *table_A, table_pred_A* populate the `from` clause. If *column_pred_A=value_pred_A, column_pred_B=value_pred_B* exist, they go into the `where` field.

The approach above is simple but effective; nevertheless two important observations are: firstly, it does not benefit from the full expressiveness of the XPath queries supported by the XMAP framework, and secondly, it requires the join conditions between tables *table_A, table_pred_A* to be inserted in a post-processing step.

Apparently, this is not the only change envisaged to the current querying services, as these are provided by OGSA-DQP. An enumeration of such modifications appears in [10].

7. Implementation Roadmap: Service Interactions and System Design

In this section we will describe in brief the system design that we envisage along with the service interactions involved.

The XMAP query reformulation algorithm is deployed as a stand-alone service, called *Grid Data Integration service (GDI)*. The *GDI* is deployed at each site participating in a dynamic database federation and has a mechanism to load local mapping information. Following the Globus Toolkit 4 [1] terminology, it implements additional *portTypes*, among which the *Query Reformulation Algorithm (QRA)* portType, which accepts XPath expressions, applies the XMAP algorithm to them, and returns the results. A database can join the system as in OGSA-DQP: registering itself in a registry and informing the *GDQS*. The only difference is that, given the assumptions above, it should be associated with both a *GQES* and a *GDI*.

Also, there is one *GQES* per site to evaluate (sub)queries, and at least one *GDQS*. As in classical OGSA-DQP scenarios, the *GDQS* contains a view of the schemas of the participating data resources, and a list of the computational resources that are available. The users interact only with this service from a client application that need not be exposed as a service.

8. Summary

The contribution of this work is the proposal of a framework and a methodology that combines a data integration approach with existing grid services (e.g., OGSA-DQP) for querying distributed databases. This way we provide an enhanced, data integration-enabled service middleware supporting distributed query processing.

The data integration approach is based upon the XMAP framework that takes into account the semantic and syntactic heterogeneity of different data sources, and provides a recursive query reformulation algorithm. The Grid services used as a basis are the outcome of the OGSA-DAI/DQP projects, which have paved the way towards uniform access and combination of distributed databases. In summary, in this paper (i) we provided an overview of XMAP and existing querying services, (ii) we showed how they can be used together through an example, (iii) we presented a service-oriented architecture to this end and (iv) we discussed how the proposed architecture will be implemented.

Acknowledgments

This research work was carried out jointly within the CoreGRID Network of Excellence founded by the European Commission's IST Programme under grant FP6-004265.

References

[1] The Globus toolkit, http://www.globus.org.

[2] M. Nedim Alpdemir, Arijit Mukherjee, Anastasios Gounaris, Norman W. Paton, Paul Watson, Alvaro A. A. Fernandes, and Desmond J. Fitzgerald. OGSA-DQP: A service for distributed querying on the grid. In *Advances in Database Technology - EDBT 2004, 9th International Conference on Extending Database Technology*, pages 858–861, March 2004.

[3] Mario Antonioletti and et al. OGSA-DAI: Two years on. In *Global Grid Forum 10 — Data Area Workshop*, March 2004.

[4] Philip A. Bernstein, Fausto Giunchiglia, Anastasios Kementsietsidis, John Mylopoulos, Luciano Serafini, and Ilya Zaihrayeu. Data management for peer-to-peer computing : A vision. In *Proceedings of the 5th International Workshop on the Web and Databases (WebDB 2002)*, pages 89–94, June 2002.

[5] Kevin S. Beyer, Roberta Cochrane, Vanja Josifovski, Jim Kleewein, George Lapis, Guy M. Lohman, Bob Lyle, Fatma Ozcan, Hamid Pirahesh, Norman Seemann, Tuong C. Truong, Bert Van der Linden, Brian Vickery, and Chun Zhang. System rx: One part relational, one part xml. In *SIGMOD Conference 2005*, pages 347–358, 2005.

[6] P. Brezany, A. Woehrer, and A. M. Tjoa. Novel mediator architectures for grid information systems. *Journal for Future Generation Computer Systems - Grid Computing: Theory, Methods and Applications.*, 21(1):107–114, 2005.

[7] Diego Calvanese, Elio Damaggio, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Semantic data integration in P2P systems. In *Proceedings of the First*

International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P), pages 77–90, September 2003.

[8] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Riccardo Rosati, and Guido Vetere. Hyper: A framework for peer-to-peer data integration on grids. In *Proc. of the Int. Conference on Semantics of a Networked World: Semantics for Grid Databases (ICSNW 2004)*, volume 3226 of *Lecture Notes in Computer Science*, pages 144–157, 2004.

[9] C. Comito and D. Talia. Xml data integration in ogsa grids. In *Proc. of the First International Workshop on Data Management in Grids (DMG05). In conjuction with VLDB 2005*, volume 3836 of *Lecture Notes in Computer Science*, pages 4–15. Springer Verlag, September 2005.

[10] Carmela Comito, Domenico Talia, Anastasios Gounaris, and Rizos Sakellariou. Data integration and query reformulation in service-based grids: Architecture and roadmap. Technical Report CoreGrid TR-0013, Institute on Knowledge and Data Management, 2005.

[11] Karl Czajkowski and et al. The WS-resource framework version 1.0. The Globus Alliance, Draft, March 2004. `http://www.globus.org/wsrf/specs/ws-wsrf.pdf`.

[12] Wenfei Fan, Jeffrey Xu Yu, Hongjun Lu, and Jianhua Lu. Query translation from xpath to sql in the presence of recursive dtds. In *VLDB Conference 2005*, 2005.

[13] Enrico Franconi, Gabriel M. Kuper, Andrei Lopatenko, and Luciano Serafini. A robust logical and computational characterisation of peer-to-peer database systems. In *Proceedings of the First International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P)*, pages 64–76, September 2003.

[14] Alon Y. Halevy, Dan Suciu, Igor Tatarinov, and Zachary G. Ives. Schema mediation in peer data management systems. In *Proceedings of the 19th International Conference on Data Engineering*, pages 505–516, March 2003.

[15] Anastasios Kementsietsidis, Marcelo Arenas, and Renée J. Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 325–336, June 2003.

[16] George Lapis. Xml and relational storage - are they mutually exclusive? available at http://www.idealliance.org/proceedings/xtech05/papers/02-05-01/ (accessed in july 2005).

[17] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 233–246, June 2002.

[18] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of 22th International Conference on Very Large Data Bases (VLDB'96)*, pages 251–262, September 1996.

[19] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.