SERVICE CHOREOGRAPHY FOR DATA INTEGRATION ON THE GRID

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

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

Abstract To date there have been several efforts with a view to developing services that support and enable data integration on the Grid; however there is a lack of a comprehensive solution to this issue. This paper summarises the work thus far on the XMAP data integration framework and query reformulation algorithm and on middleware with regard to Grid query processing services, namely OGSA-DQP. Furthermore, it presents an architecture for data integration-enabled query processing on the Grid, which combines the two aforementioned pieces of work and provides an extended set of e-Services. These services allow users to submit queries over a single database and receive the results from multiple databases that are semantically correlated with the former one. The paper focuses on the service choreography involved by elaborating on the interactions between the services, and discusses the extensions to OGSA-DQP that are required in order to make the services interoperable.

Keywords: service choreography, data integration, XMAP, OGSA-DQP.

1. Introduction

The Grid, as an emerging infrastructure for the discovery, access and use of distributed computational resources [15], offers new opportunities and raises new challenges in data management. Many aspects differentiate the Grid from a traditional distributed environment; such aspects include 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 issue in managing data on Grids is reconciliation of data heterogeneity [8]. 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 one of the most persistent problems that the database and information management community has to deal with. Although significant progress has been made in several aspects of data integration, the increase in availability of web-based data sources has led to new challenges. More specifically, efficient techniques have been developed and approaches have been devised to schema mediation languages, query answering algorithms, optimisation strategies, query execution policies, industrial development, and so on [17]. However, effective techniques for the generation and handling of semantic mappings are still in their infancy. The need for semantic correlation of data sources is particularly felt in Grid settings. Moreoever, in a Grid, a centralized structure for coordinating all the nodes may not be practical because it can become a bottleneck and, more importantly, it cannot accommodate the dynamic and distributed nature of Grid resources.

Data access and integration services have been attracting significant interest from the Grid community. Data Grids that rely on the coordinated sharing of and interaction across multiple autonomous database management systems play a key role in many industrial and scientific initiatives. To this end, middleware services have been developed. Two notable examples are the *OGSA Data Access and Integration* (OGSA-DAI) [6] and the *OGSA Distributed Query Processor* (OGSA-DQP)[1] [5, 4] projects. These projects have moved toward a servide-oriented architecture quite early in their lifecycle. OGSA-DAI exposes database management systems (including Oracle, MySQL, SQLServer, DB2, and so on) in a uniform way, whereas OGSA-DQP provides distributed query processing functionalities on top of OGSA-DAI. As such, OGSA-DQP can combine and integrate data from multiple data sources. To enhance performance, it employs parallel query execution techniques; nevertheless it relies

[1] OGSA-DQP is publicly available in open source form from www.ogsadai.org.uk/dqp.

on the user for the semantic interpretation of the data and does not address any schema integration requirements.

To date, only few projects (e.g., [11, 9]) actually meet the schema-integration requirements that are necessary for establishing semantic connections among heterogeneous data sources. To address this limitation, the use of the *XMAP* framework for integrating heterogeneous data sources distributed over a Grid has been proposed [12] . The aim of this framework is to develop a decentralized network of semantically related schemas, so that the formulation of distributed queries over heterogeneous data sources is enabled. XMAP employs a decentralized point-to-point mediation approach to connect different data sources based on schema mappings in order to combine remote XML documents. The XMAP framework is also exposed as an additional e-Service, called *Grid Data Integration Service* (GDIS). The contribution of the paper is the presentation of a unifying infrastructure for distributed query processing and query reformulation driven by semantic connections. The infrastructure proposed exploits the middleware provided by OGSA-DQP and OGSA-DAI, to provide schema-integration services. The integration and coordination of different services is the topic of service choreography. In this paper, we examine in detail how OGSA-DAI/DQP and GDIS are combined.

The remainder of the paper is organized as follows. Section 2 presents a short analysis of data integration systems focusing on the issues that are more relevant to Grids. The integrative architecture that combines the query reformulation and the query processing services, along with their interaction, is the subject of Section 3. Section 4 presents the XMAP integration framework, and describes the underlying integration model and the XMAP query reformulation algorithm. Section 5 discusses a simple example of applying the XMAP algorithm to OGSA-DQP supported relational databases, elaborating on how the service integration is achieved in practice and how the architecture proposed can be further extended. Finally, Section 6 concludes the paper.

2. Background

Both areas of data integration and Grid computing benefit from their combination:

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

- integration formalisms can benefit from an OGSA-based Grid infrastructure, since such an infrastructure facilitates dynamic discovery, allocation, access, and use of both data and computational resources, which are 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 federation database management systems (FDBMS) [22] and the use of mediator/wrapper middleware [21], are not suitable in Grid settings.

The federation approach is a rather rigid configuration where resource 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 is performed by a central administrator, which is an obstacle to the exploitation of evolving characteristics of dynamic environments. As a consequence, these approaches are insufficient when data sources change often and to a significant extent, since such changes may violate the mappings to the mediated schema.

Recently, several works on data management in peer-to-peer (P2P) systems are moving towards decentralized, wide-scale sharing of semantically-related data [7, 10, 16, 18, 19]. All these systems focus on an integration approach, which is not based on a global schema: each peer represents an autonomous information system, and data integration is achieved by establishing mappings between 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* [11] and *GDMS* [9]. Both systems are based on an approach similar to ours, i.e., to build data integration services by extending the reference implementation of OGSA-DAI. The *Grid Data Mediation Service* (GDMS) is part of the GridMiner project [16] and 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. The main difference from our work is that it relies on the existence of a global schema, which is not that realistic in Grids. *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. In that framework, the authors use a simple relational language for expressing both the schemas and the mappings. Our integration model follows an approach not based on a hierarchical structure as well, however it focuses on XML data sources and is based on schema-mappings that associate paths in different schemas. Finally, semantic mapping across relational databases coupled with a global-as-view approach is investigated in the context of the SASF project [3].

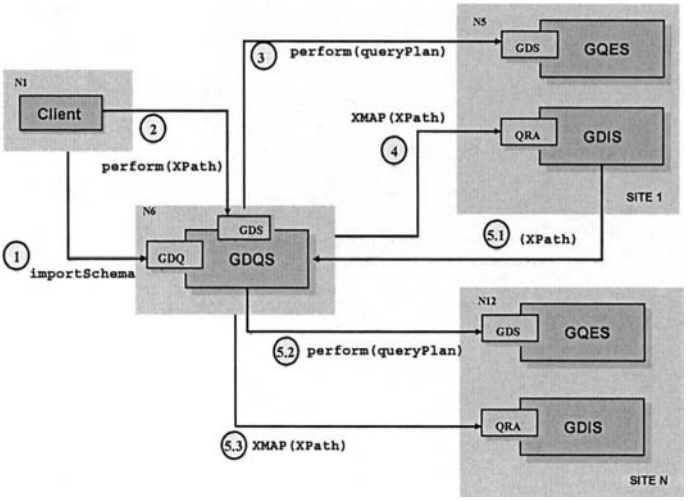

Figure 1. Data integration-enabled query processing on the Grid: service interactions.

3. Architecture and Service Interactions

The XMAP query reformulation algorithm, presented in more detail in the following section, is deployed as a stand-alone service, called *Grid Data Integration Service (GDIS)*. Given an XPath query over a local database, it returns the equivalent XPath queries that retrieve semantically similar data from remote databases. Figure 1 provides an overview of the service interactions involved in the incorporation of data integration functionality in distributed query processing on the Grid. It focuses on the interactions that concern the *GDIS*, and thus it hides all the complexities that relate to (distributed) query submission and execution. As such, it complements the service interactions between the OGSA-DAI and DQP services.

OGSA-DQP is an open source service-based Distributed Query Processor; as such, it supports the evaluation of queries over collections of potentially remote data access and analysis services. OGSA-DQP uses Grid Data Services provided by OGSA-DAI to hide data source heterogeneities and ensure consistent access to data and metadata from any database resource. The current version of OGSA-DQP, OGSA-DQP 3.0 is Globus Toolkit 4 compliant [9]. Thus OGSA-DQP builds upon the WSRF infrastructure.

OGSA-DQP provides two additional types of services, *Grid Distributed Query Services (GDQSs)* and *Grid Query Evaluation Services (GQESs)* or simply *Query Evaluation Services (QESs)*. The former are visible to end users, 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 OGSA-DAI services, ordinary Web Services

and other instances of GQESs), and are responsible for the execution of the query plans created by a GDQS. The interactions and functionality of OGSA-DQP services are described in detail in [4]. In the latest OGSA-DQP version, GQESs have been refactored as ordinary Web Services, augmenting the applicability of OGSA-DQP, as its deployment has been simplified significantly, whereas several interdependencies have been removed.

For the unifying architecture, the following architectural assumptions are made. A *GDIS* is deployed at each site participating in a dynamic database federation and has a mechanism to load local mapping information. It implements an additional *portType*, namely *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 *GDIS*.

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 a GDQS service through a client application that need not be exposed as a service.

A comprehensive data integration architecture needs to combine both the query reformulation and the query processing services. The interactions of the services, which form the choreography for data integration, are as follows (see also Figure 1):

1 The client contacts the *GDQS* and requests a view of the schema for each database he/she is interested in. At this point, there is no assumption that the user has an a-priori knowledge of the semantics of this and the semantically-related databases.

2 Based on the retrieved schema, he/she composes an XPath query, which is sent to the *GDQS*, and not directly to the corresponding database service, following the OGSA-DQP approach.

3 The *GDQS* transforms, parses, optimises, schedules and compiles a query execution plan [23]. This process entails the identification of the relevant sites, and consequently their local *GQES* and *GDIS*. The resulting query execution plan is sent to the corresponding *GQES*, which returns the results asynchronously, after contacting the local database via an OGSA-DAI service.

4 The initial XPath expression is sent to the *GDIS* that is co-located with the *GQES* of the previous step to perform the XMAP algorithm. *GDIS*

retrieves the locally stored mapping schema, which contains the mapping information that links the paths in the submitted query with paths referring to other databases.

5 As long as the call to the *GDIS* returns at least one XPath expression that has not been considered yet in the same session, the following steps are executed in an iterative manner.

(a) The results of the call to the *GDIS*, which contain a set of XPath expressions, are collected by the *GDQS*. Subsequently, the *GDQS* filters out the ones that have already been processed in the current session.

(b) Each remaining XPath expression is processed as in Step 3 to collect results from databases other than the one initially considered.

(c) The same XPath expressions are processed as in Step 4 to find additional correlated queries. I.e. there is a loop which continuously generates XPath queries until all the relevant data has been retrieved.

4. The XMAP Integration Framework

The primary design goal of 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 modelled 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 [12] 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 [12], we propose a decentralized approach inspired from [18] 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 map-

ping 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 utilizes data from any node that is transitively connected by semantic mappings, and reformulates the given query expanding and translating it into appropriate, equivalent 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.

Figure 2 shows the service interface of the *Grid Data Integration Service*, which defines *Query Reformulation Algorithm (QRA)* portType. Such a service is interoperable with any other common Web and Grid Services.

5. Combining query processing and reformulation services

The XMAP algorithm can be used for data integration-enabled query processing in OGSA-DQP. The example discussed in this section 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.

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 [12]. Initially, the users submit XPath queries that refer to a single physical database. E.g., `/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, `/S1/Artefact/title` returns the titles of the artifacts in the same database. When the XMAP algorithm is applied for the second query, two

```
<?xml version="1.0"?> <!-- root element wsdl:definitions defines
set of related services-->

<wsdl:definitions name="QueryReformulation"
  xmlns:qr="http://.../QueryReformulation.wsdl"
  xmlns:qrxsd="http://.../QueryReformulation.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
<wsdl:types>
 <xsd:schema targetNamespace="..."
      xmlns:xsd="http://www.w3.org/1999/XMLSchema">
   <xsd:element name="ArrayOfString">
    <xsd:complexType >
     <xsd:sequence>
      <xsd:element name="XPathQuery"
         type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
 </xsd:schema>
</wsdl:types> <wsdl:message name="queryToReformulate" >
 <wsdl:part name="inputQuery" element="xsd:string"/>
</wsdl:message> <wsdl:message name="reformulatedQueries" >
 <wsdl:part name="reformulatedQuery" element="qrxsd:ArrayOfString"/>
</wsdl:message> <wsdl:portType name="QRAPortType">
  <wsdl:operation name="reformulation">
     <wsdl: input message="qr:queryToReformuate/>
     <wsdl: input message="qr:reformulatedQueries/>
 <wsdl:operation/>
<wsdl:portType/> <wsdl:binding
name="QueryReformulationSoapBinding" type="qr:QRAPortType">
 <soap:binding style="document"
              transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="reformulation">
      <soap:operation soapAction="..."/>
       <wsdl:input>
        <soap:body use="literal"  namespace="..."/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" namespace="..."/>
      </wsdl:output>
      <wsdl:fault>
        <soap:body use="literal"  namespace="..."/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="QueryReformulationService">
    <wsdl:documentation>...</wsdl:documentation>
    <wsdl:port name="QRAPortType"
              binding="qr:QueryReformulationSoapBinding">
      <soap:address location="..."/>
    </wsdl:port>
  </wsdl:service>
 </wsdl:definitions>
```

Figure 2. The interface of the Grid Data Integration Service.

more XPath expressions will be created that refer to the *S2* database: `/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;,
```

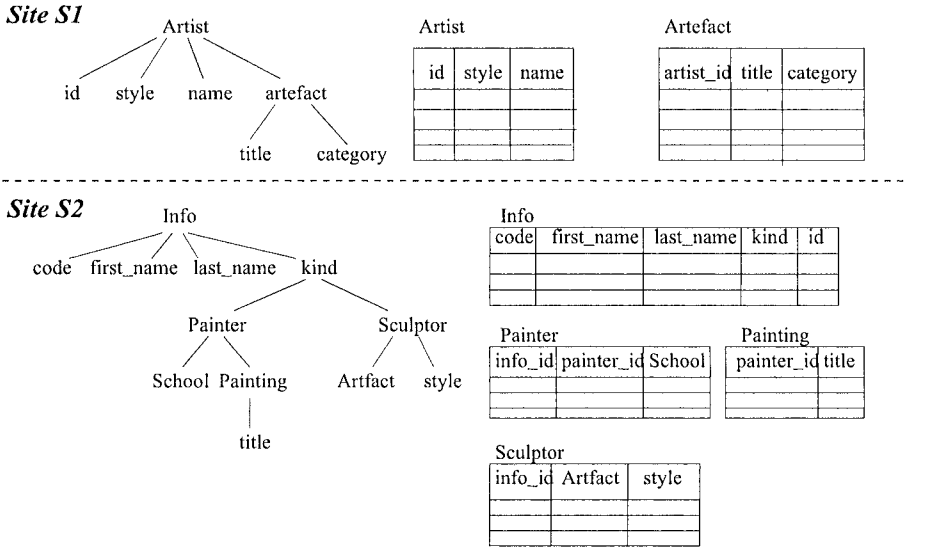

Figure 3. The example schemas.

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

Note that the mapping of simple XPath expressions to SQL/OQL is feasible [20]. However, solving the mismatch between OQL and XPath, is not the only problem. The grid querying services provided from OGSA-DQP cannot support the proposal of this paper as they are. The modifications required are presented in more detail in the following subsection.

5.1 A Summary of the Extensions Envisaged to the Current Querying Services

The afore-mentioned architecture, apart from the development of the new *GDIS* service, implies some extensions to the current services and clients that are available from OGSA-DAI and OGSA-DQP. These extensions are, in our view, reasonable and feasible, and thus make the overall proposal of practical interest. They are summarised below:

- Currently, *GDQS* does not reveal any information on the database to which a table belongs, as the purpose of OGSA-DQP is to present a unified view of all the database schemas to the user hiding the locality details. However, in the proposed architecture, the user requirements change and the queries are submitted to a single physical database. As such, the client should expose the schemas per database rather than as a

```
<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.

```
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.

unified view, so that it becomes evident what exactly data each database holds.

- *GDQS* should be capable of accepting XPath queries, and of transforming these XPath queries to OQL before parsing, compiling, optimizing and scheduling them. Such a transformation falls in an active research area (e.g., [14, 8]), and, in our architecture, is realised as an additional component within the query compiler.
- *GDQS* should implement an additional XMAP-related activity that, given an XPath expression, finds the corresponding *GDIS*, and calls the XMAP on it. The activity returns a set of corresponding XPaths.
- The client should be capable of aggregating results stemming from multiple queries.
- *GDQS* should be capable of accepting requests that contain more than one (XPath) statement.
- Also, *GDIS* should be capable of processing requests that clean, update and install mapping documents.

5.2 Looking Ahead

The proposed architecture provides added value to the existing querying services, and increases the scope of the applications that may use them. It creates a middleware infrastructure that can be enhanced with more functionality. With a view to incorporating more features, the following stages of extensions have been identified:

Stage A: XPath is a simple language, and, as such, it cannot cover many of the common user requests. It is expected that more extensive use of the knowledge about key/foreign-key relationships will be required in order to reformulate more expressive queries (such as XQuery, SQL and OQL correctly) or to support a more complete set of XPath. When the paths in a XPath query refer to different branches of the tree of the corresponding XML document, the relevant OQL/SQL query typically contains a

join, as it is more convenient to map such branches to distinct relational tables. However, the join condition is implied and cannot be directly derived from the XPath expression. Consequently, the knowledge of the key/foreign-key between tables is essential for the correct reformulation of a wider range of XPath queries in our proposal.

Stage B: OGSA-DQP naturally provides the capability to submit queries over distributed sources in a manner that is transparent to the user. The XMAP reformulation algorithm, as presented in [12], returns a new query only if that query can be evaluated across a single database as well; this is one of the validity criteria. In order to use the capability of OGSA-DQP to evaluate distributed queries across multiple databases in the future, some (non-extensive) changes in the validity criteria of reformulated queries in the XMAP algorithm will be required.

Stage C: A more challenging problem is to allow distributed query reformulation. This raises a new set of issues, which include the selection of the site that should hold the mappings, the identification of any further metadata at the GDQS-level that is required, and ensuring that non-duplicate results are produced. More specifically, in the proposed architecture, the decision on which *GDI* Service should perform the query reformulation is straightforward; it is the one that is co-located with the database that holds the data retrieved by the relevant query, and this service contains the full set of the mapping information required. However, when multiple databases are accessed in the same query, this policy has to be revised.

Stage D: Finally, we plan to explore alternative architectures, and especially architectures in which the *GDISs* may not be co-located with *GQESs*, and can be shared between multiple sites. A simple approach could be to have a single *GDI* Service that contains the full mapping information concerning all the semantically similar databases. However, such an approach is not scalable. If there are multiple *GDISs*, which are not co-located with *GQESs*, then a co-ordination issue arises as to how (i.e., according to which protocol) the services interact and exchange knowledge. To this end, adopting techniques from peer-to-peer models is a promising strategy.

6. Conclusions

The contribution of this work is the proposal of a unifying architecture and of an approach that integrates a data integration methodology with existing e-Services for querying distributed databases with a view to providing an enhanced, data integration-enabled service middleware. The resulting architec-

ture remains service-oriented, and, as such, the service choreography issues are important. The paper explains in detail how the distinct services can interact in order to accomplish the non-trivial task of evaluating remote queries submitted by the user, while, at the same generating automatically new queries that return semantically similar results from different data sources. The data integration is based upon the XMAP framework that takes into account the semantic and syntactic heterogeneity between different data resources, 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 propose an integrated service-oriented architecture; (ii) we explain how we can achieve interaction between the various services; (iii) we show how these services can be used together through an example; (iv) we discuss in detail the implementation issues involved; and (v) finally, we provide insights into how the architecture can be further extended.

Acknowledgments

The collaboration between the Universities of Manchester and Calabria is supported through the CoreGrid European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies project. We are pleased to acknowledge their support.

References

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

[2] GridMiner, http://www.gridminer.org/.

[3] SASF: service-based approach to schema federation, http://sasf.grid.leena34.com/.

[4] M. N. Alpdemir, A. Mukherjee, N. W. Paton, P. Watson, A. A. Fernandes, A. Gounaris and J. Smith. Service-based distributed querying on the grid. In Maria E. Orlowska, Sanjiva Weerawarana, Mike P. Papazoglou, and Jian Yang, editors, *Service-Oriented Computing - ICSOC 2003, First Int. Conference, Trento, Italy, December 15-18, 2003, Proceedings*, pages 467–482. Springer, 2003.

[5] M. N. Alpdemir, A. Mukherjee, A. Gounaris, N. W. Paton, P. Watson, A. A. Fernandes and D. J. Fitzgerald. OGSA-DQP: A service for distributed querying on the grid. In *Advances in Database Technology - EDBT 2004, 9th Int. Conference on Extending Database Technology*, pages 858–861, March 2004.

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

[7] Ph. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini and I. Zaihrayeu. Data management for peer-to-peer computing : A vision. In *Proc. of the 5th Int. Workshop on the Web and Databases (WebDB 2002)*, pages 89–94, June 2002.

[8] K. S. Beyer, R. Cochrane, V. Josifovski, J. Kleewein, G. Lapis, G. M. Lohman, B. Lyle, F. Ozcan, H. Pirahesh, N. Seemann, T. C. Truong, B. Van der Linden, B. Vickery and

C. Zhang. System rx: One part relational, one part xml. In *SIGMOD Conference 2005*, pages 347–358, 2005.

[9] P. Brezany, A. Woehrer and A. M. Tjoa. Novel mediator architectures for grid information systems. *Future Generation Computer Systems*, 21(1):107–114, 2005.

[10] D. Calvanese, E. Damaggio, G. De Giacomo, M. Lenzerini and R. Rosati. Semantic data integration in P2P systems. In *Proc. of the First Int. Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P)*, pages 77–90, September 2003.

[11] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati and G. 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.

[12] C. Comito and D. Talia. Xml data integration in ogsa grids. In *1st Int. Workshop on Data Management in Grids (DMG)*, pages 4–15, 2005.

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

[14] W. Fan, J. Xu Yu, H. Lu and J. Lu. Query translation from xpath to sql in the presence of recursive dtds. In *VLDB Conference 2005*, 2005.

[15] I. Foster, C. Kesselman and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Int. J. Supercomputer Applications*, 15(3), 2001.

[16] E. Franconi, G. M. Kuper, A. Lopatenko and L. Serafini. A robust logical and computational characterisation of peer-to-peer database systems. In *Proc. of the First Int. Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P)*, pages 64–76, September 2003.

[17] A. Y. Halevy. Data integration: A status report. In *BTW*, pages 24–29, 2003.

[18] A. Y. Halevy, D. Suciu, I. Tatarinov and Z. G. Ives. Schema mediation in peer data management systems. In *Proc. of the 19th Int. Conference on Data Engineering*, pages 505–516, March 2003.

[19] A. Kementsietsidis, M. Arenas and R. J. Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In *Proc. of the 2003 ACM SIGMOD Int. Conference on Management of Data*, pages 325–336, June 2003.

[20] G. 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).

[21] A. Y. Levy, A. Rajaraman and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of 22th Int. Conference on Very Large Data Bases (VLDB'96)*, pages 251–262, September 1996.

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

[23] J. Smith, A. Gounaris, P. Watson, N. W. Paton, A. A. Fernandes and R. Sakellariou. Distributed query processing on the grid. In Manish Parashar, editor, *Grid Computing - GRID 2002, Third Int. Workshop, Baltimore, MD, USA, November 18, 2002, Proceedings*, pages 279–290. Springer, 2002.