Virtualization of Heterogeneous Data Sources for Grid Information Systems

Alexander Wöhrer and Peter Brezany and Ivan Janciak
Institute for Software Science
University of Vienna
Liechtensteinstrasse 22, A-1090 Vienna, Austria
Email: {brezany|janciak|woehrer}@par.univie.ac.at

***Abstract*— Grid Information Systems will use *existing data* from *various distributed and heterogeneous data stores* as well as new data entering the organization. Several technical obstacles arise in the design and implementation of a system for integration of such data sources – most notably *distribution, autonomy*, and *data heterogeneity*. This paper describes the data integration system based on the *wrapper-mediator approach* – namely the *Grid Data Mediation Service* – of the *GridMiner* project conducted in Vienna. The developed mediation service is, to our best knowledge, the *first prototype* of a Grid service capable of presenting distributed, heterogeneous data sources as one logical *virtual data source* on the Grid. We developed a flexible *mapping schema* to describe the building process of a virtual data source. At present, integratable data sources include structured as well as semi-structured data sources. Although we are (currently) not describing the semantics of the data sources, our system permits the possibility to include *own Java transformation functions* (static and dynamic) in the mediation process to resolve all kinds of heterogeneities.**

I. Introduction

The Grid [1] is the computing and data management infrastructure, which is transforming science, business, health and society. The original idea behind the Grid was not to offer an alternative to the Internet, or to rival it in any way, but more to complement its functionality, and to add flexibility. The Grid can be viewed as an extension to the Web, building on its protocols, and offering new functionality. At its core, the Grid is all about *distributed computing and resource management*. The need for open standards that define interaction and encourage interoperability between components supplied from different sources was the motivation for the Open Grid Services Architecture (OGSA) [2].

One reason why Grid computing gains importance is the *changed historical trade-off* [3] between networking and processing costs. Another reason is that today, a vast amount of potential computing capacity remains *untapped*. The truth is, *flexibility* is a key issue. Grid Computing allows the system or network to be *virtualised* so that it can be dynamically reconfigured in different ways as business requirements change. This *virtualization* of resources places all of the necessary access, data and processing power at the fingertips of those who need to rapidly solve complex business problems, conduct compute-intensive research and data analysis, and engage in real-time.

The analysis of the requirements of e-Science projects [4] has shown that there is an urgent and widespread need for the interconnection of pre-existing and independently operated databases. In many large companies, the widespread usage of computers has also led a number of different application-specific databases. As company structures evolve, boundaries between departments move, creating new business units. Their new applications will use existing data from various data stores, rather than new data entering the organization. Henceforth, the ability to make data stores interoperable becomes a crucial factor for the development of new information systems. However, if the Grid is to support a wider range of applications, both scientific and commercial, then database integration into the Grid will become important. Therefore, the Global Grid Forum Database Access and Integration Services (DAIS) [5] Working Group developed a specification for a Grid database services. The first reference implementation of this specification, OGSA-DAI Release 3 [6], is already available.

The next logical step is the support for federating data resources, as depicted in Fig. 1, which is vital to the success of the Grid. The alternative of forcing each application to interface directly to a set of databases and resolve federation problems internally would lead to application complexity, and duplication of effort. The factors that make Grid database federation different include for example

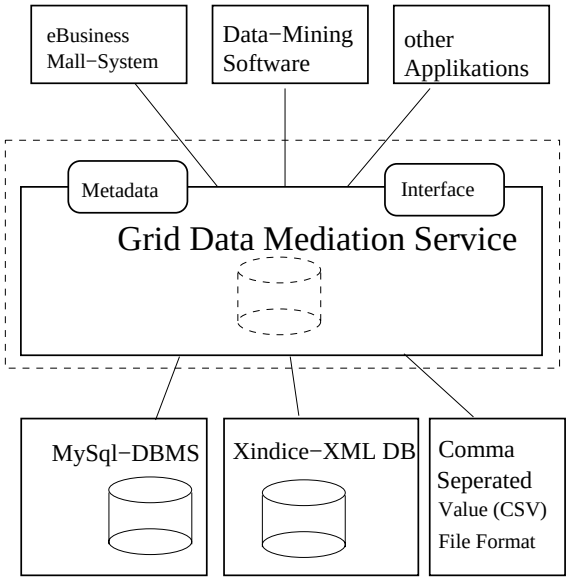

Fig. 1. Grid DataMediation Service (GDMS) providing a virtual data source (VDS) for different applications

high dynamic flexibility, extreme performance, and semantic aspects.

This paper describes the data integration system based on the *wrapper-mediator approach* [7] – namely the *Grid Data Mediation Service* (GDMS) [8] – of the *GridMiner* project [9] conducted in Vienna. This project represents the first research effort to build a novel, service oriented *infrastructure for knowledge discovery* in datasets (KDD) integrated into Computational Grids based on the OGSA-based Globus Toolkit [10]. The developed mediation service is, to our best knowledge, the *first prototype* of a Grid service capable of presenting distributed, heterogeneous data sources as one logical *virtual data source* on the Grid. The developed concepts have been implemented (but are not limited to) by re-using the OGSA-DAI Grid Data Service as a framework to show their feasibility. This contribution significantly leverages the functionality of the OGSA-DAI reference Grid Data Service implementation by enabling to access *more than one* data source over a global schema with a subset of SQL[1] and integrating the results in a standardised way. For first performance results see [8].

The remaining part of the paper is organized as follows. Section II is illustrating the architecture and implementation of the GDMS on the example of an mediation process for a data mining task. Section III discusses related work whereas Section IV briefly outline the future work. The paper is closed with our conclusions in Section V.

[1]Structured Query Language

II. Grid Data Mediation Service

As mentioned earlier, in order to avoid building a new proprietary solution and reimplementing well solved aspects of GDSs, we have decided to integrate our developed concepts into the free available OGSA-DAI Grid Data Service (GDS) reference implementation to provide a virtual data source (VDS). We implemented the prototype of a Grid Data Mediation Service (GDMS) by reusing the framework provided by OGSA-DAI. The GDS is the *primary OGSA-DAI service* providing *grid-enabled access to one single data source at the time* using a *document-oriented model* in form of an XML document.

We are explaining the implementation of our GDMS with the following example *access scenario* from the domain of health care. Let's suppose you want to mine data sources from different sites as depicted in Fig. 2. The data of the two involved hospitals is distributed over the three departments A, B and C. Although the two hospitals store the same information about their patients, the data structures are different.

Let's assume that the name information of hospital one is represented by the patient's full name, and the name information of an patient at hospital two is divided into first name (fn) and last name (ln). Hospital one has *no central* patient database and so the data is divided and stored in two different data sources - the administrative info in the administration department data source (in Fig. 2 called site A) and the medical information in the care unit data source (in Fig. 2 called site B). The other informations provided like date of first treatment, day of birth and adress have the same structure but are accessible over different names. To *materialize* the virtual data source (i.e. to reconstruct it from its fragments), the following operations are required: *R = (A JOIN B) UNION C*.

Before we can start to mediate, we clearly need the information how the VDS is built (what data sources are participating and how they are combined) and how it looks like to the user (the *mediated schema*). Our GDMS uses a *mapping schema* – which defines mapping information between the virtual data source and the participating data sources – for the integration of the results from the different data sources. For the *mediated schema* we are utilising the same schema as a conventional

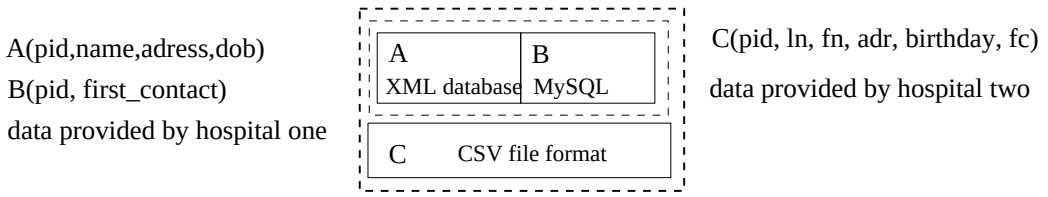

Fig. 2. Fragmentation of the virtual data source over three heterogeneous data sources

GDS representing one relational database. Users pose queries in terms of the mediated schema rather than directly in terms of the source schemas. As such, the mediated schema is a set of *virtual relations*, in the sense that they are not actually stored anywhere (virtual integration approach).

The XML instance outlined in Fig. 3 is an example for a mapping schema in order to mediate between the data resources given in our example scenario. The main element *VDSConfig* consists of a subelement *VDSTable* which represents the mapping information for each virtual table contained in the *VDS*. This subelement contains other subelements such as *union*, *select* and *join* – all named via the *result_name* attribute.

Union operations are defined by enumerating the fragments to merge in the *union* element, which are in our example the results of a join operation (between A and B) and the results of a selection (from C).

The *joinInfo* element contains the information required for the join operation, including the kind of the join given by the attribute *kind* and the key columns needed. In our example the key column is *p_id* and we want to perform an inner-join.

The *select* elements are responsible for accessing the various data sources participating in the VDS[2]. The attribute *dataResource-Ref* specifies the location (connection URI) of the data source, whereas the *select-Impl* attribute determinates the Java class[3] to use to query the data source. The *source-part-ref*-element specifies on which table or collection[4] the *mapSource* definitions have to be applied.

Fig. 4(a) shows the instance of a *mapSource* element applied to our example to query the *CSV file*. First the column seperator (via the *ColSeperator* element) and the line seperator (via the *LineSeperator* element) are defined. The following *column* elements describe how the logical names of the

```
<VDSConfig>
 ...
 <VDSTable table_name="patient">
  <union result_name="U" kind="all">
   <join result_name="J">
    <select result_name="A"
            dataResource-Ref="xmldb:..."
            select-Impl="...">
     <mapSource>
     ...
     </mapSource>
     <source-part-ref/>
    </select>
    <select result_name="B"
            dataResource-Ref="jdbc:..."
            select-Impl="...">
     <mapSource>
     ...
     </mapSource>
     <source-part-ref>...</source-part-ref>
    </select>
    <joinInfo kind="inner">
     <left_part key_col_list="pid"/>
     <right_part key_col_list="pid"/>
    </joinInfo>
   </join>
   <select result_name="C"
           dataResource-Ref="file://..."
           select-Impl="...">
    <mapSource>
    ...
    </mapSource>
    <source-part-ref>...</source-part-ref>
   </select>
  </union>
 </VDSTable>
 ...
<VDSConfig>
```

Fig. 3. Example mapping schema for a table in the VDS

VDS table are built from data in the CSV file. For the logical name[5] *p_name* we need to know how to combine it from the two seperat values *ln* (lastname) and *fn* (firstname) – this is done by using our *transformation function concept*.

This concept is our mechanism for resolving semantic and representation heterogenities between the data. It uses the possibility of the *XQuery engine* from SAXON [11] to include your own

[2]so far comma separated files, relational databases via JDBC and Xindice databases

[3]concrete wrapper implementation for one kind of sources

[4]part of an XML database

[5]the ones specified in the mediated schema

```
<mapSource>
 <ColSeperator>;</ColSeperator>
 <LineSeperator>\r\n</LineSeperator>
  ...
  <column column-ref="p_name"
   transformStatement="combineName(fn,ln)"
   fullJavaPath="TestTransform">
   <soure-ref>2</soure-ref>
  </column>
  <column column-ref="ln">
   <soure-ref>2</soure-ref>
  </column>
  <column column-ref="fn">
   <soure-ref>3</soure-ref>
  </column>
  ...
</mapSource>
```

(a) MapSource instance to access the CSV file

```
public class TestTransform {

public static String combineName(
       String one,String two)
{
    return one+" "+two;
  }

}
```

(b) Transformation function for the mediation process

Fig. 4. MapSource instance to access a CSV file and the proper Transformation function

Java functions (static and dynamic) into a mediation process. The Java code pictured in Fig. 4(b) defines the static function needed for our example CSV file. The *mapSource* description is a very flexbile schema and can be adopted to various data sources – only the column descriptions with their corresponding logical name in the VDS table (via the *column-ref* attribute) are mandatory.

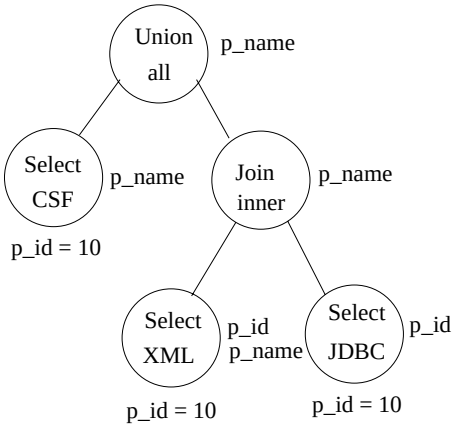

Fig. 5. Operator tree to materialize a query against the VDS

The developed mediation service is seamlessly integrated into the reference implementation of the OGSA-DAI Grid Data Service processing the provided information about the VDS described above. The overall system architecture of the GDMS is shown in Fig. 6.

Query reformulation. In the first phase, the query against the mediated schema is parsed and translated into an internal representation – a query graph as shown in Fig. 5 – that can be easily processed by the later phases. In our system, where data is distributed, this part also selects the partitions of a table that must be considered to answer a query and the applying of the conditions (if specified in the *WHERE-clause* of the query) as early as possible. Fig. 5 represents the *customized* operator tree to collect and combine the results for our example virtual table *patient* given by the mapping schema. The circles represent operators, where the first row shows the name of the operator and second row the respective implementation needed. The line under the circles show the *logical names* of the patient table which have to be returned by the operator in order to materialize the table – e.g. for a *'SELECT p_name FROM patient WHERE p_id = 10'*. After this, the query is transformed in order to carry out optimizations that are good regardless of the physical state of the system. Typical transformations are the elimination of redundant predicates or simplification of expressions. With the help of the *mapping schema* e.g. given in Fig. 3 and the mediated schema (which represents the user view), the main part of the mediation is done.

Query Optimizer. This component carries out optimizations that depend on the physical state of the system. E.g. the optimizer is able to choose another *replica* (if specified) when one replicated data resource isn't responding in order to materialize the VDS anyhow. This is possible trough working with logical names during the whole mediation process until the execution phase. For example, let's assume the required MySQL data source is not availiable but the data has been

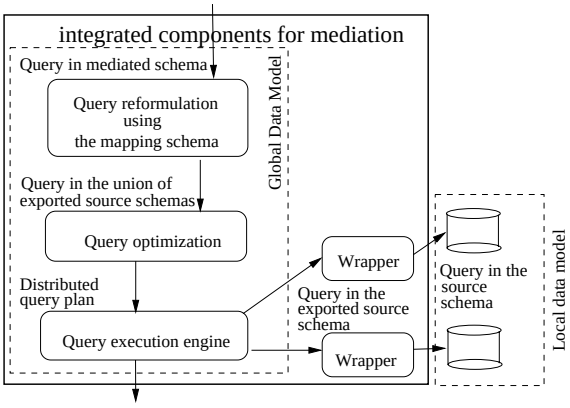

Fig. 6. Architecture of the GDMS modules integrated into OGSA-DAI

exported into a CSV file which is specified as a replica for the MySQL database. To integrate the CSV file instead of the MySQL database, only the leaf of the operator tree shown in Fig. 5 has to be re-instantiated with another implementation for the select operation – anything else stays unchanged.

Query Execution Engine. This element provides generic implementations for every operator. In our model, operators are implemented as iterators and all iterators have the same interface. As a result, any two iterators can be plugged together (as specified by the consumer producer relationship of a plan), and thus, any plan can be executed. Another advantage of the iterator model is that it supports the pipelining of results from one operator to another in order to achieve good performance. The provided operators in our system are based on the free XQuery engine SAXON [11] for fast prototyping, ease of use and adaptability to new data sources and functionalities.

III. Related Work

Basically, the work presented here addresses mediation of various heterogenous data sources over the Grid. Grid database access is being developed and researched by the Database Access and Integration Services Working Group (DAIS-WG) [5] in close cooperation with the OGSA-DAI project [6]. The OGSA-DAI Distributed Query Processing (DQP) system [12] can be seen as a mediator component over OGSA-DAI wrappers. Its main concern is efficient query execution rather than reconciliation of data source heterogeneity as done by our *GDMS* described in Section II.

Database mediation has been studied for quite a while [13], especially for distributed and heterogeneous systems [14], [15]. Recently there is a trend to use XML either to

– transport queries and/or query results, done in [6]

– describe database schemas, shown in [16]

– store the data itself, described in [17]

Optimization of query execution for parallel and distributed databases is also an extremely relevant topic [18]. Since Grid services are distributed over the Internet which mostly offers very indeterministic connections (in terms of speed of data transfer or reliability) adaptive optimization strategies that can immediately react to changes in the environment seem promising [19].

IV. Future work

A prototype implementation of the GDMS for vertical as well as horizontal partitioning has been developed, applying transformation function to resolve conventional heterogeneities, to prove the feasibility of the described concept. To increase the usability and performance of our mediation system, future work research agenda include:

Evolve the developed concepts so that they can be applied in a distributed/parallel mediation service. The mediator/wrapper approach for accessing heterogeneous data sources was originally a distributed one [7] – designed to use other mediators as components.

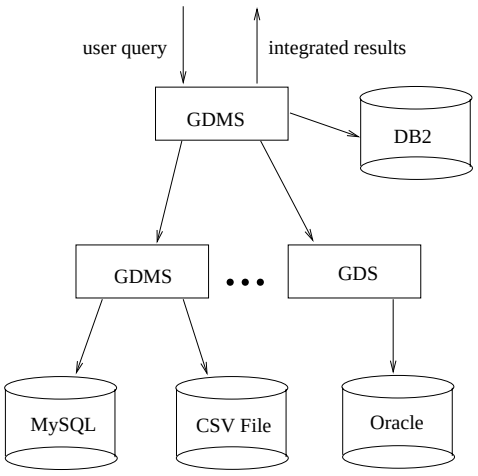

Fig. 7. Data Integration System – a distributed approach

This would have a lot of advantages, e.g. performance and maintainability. Performance would be increased by applying conditions earlier/nearer to the source – imagine the speedup for accessing

a CSV file over a GDMS located at the same machine compared to first load the whole file to a GDMS somewhere on the Grid and then only apply the conditions for the data source. Recall into memory the famous IT approach *'Divide and Conquer'* – also maintainability would be eased by allowing to split the work into smaller, easier manageable pieces. For example, each organisation could maintain their own data sources and the combination of them – where originalities and specialities of the data is known and understood. That becomes even more desirable if semantic describions should be provided for data sources.
Refine the mapping schema and with it the functionality of the system to support mediation tasks as e.g. *outer join* and *union all* operations. This could be supported by integrating the *concept of ontologies* [20]. For example describing the semantics of provided tables and their columns will add value to the mediation service and increase usability[6]. Also the context of their origin could be described with ontologies. It would also support the development of an (semi-) automatic integration process.

V. Conclusion

In this research effort, we have defined an extendable and flexible system architecture, called *Grid Data Mediation Service*, which is able to reduce complexity when accessing heterogenous and distributed data.

As the Grid is evolving from providing raw computing power to a infrastructure integrating huge amount of data from various organisations and origins, middleware which integrates and homogenises this data is *urgently needed* for the successfull development of higher level services, e.g. for data mining. The prototype implementation, which is based on XML schemes to provide the information needed to build a virtual data source, already shows the feasability of the developed concepts.

It provides a clean abstraction of heterogeneous data for users/applications, supports a wide variety of data formats and is useable with a subset of the well known SQL. By providing different views of data, according to various applications needs, it fulfills requirements for high dynamic and flexible data access. Offering the same metadata/interface as a conventional GDS, it can be used and integrated in existing applications quite easily to hide the distribution and heterogeneity of the participating data sources. Although the prototype is developed in the *GridMiner* project, its usage is *not limited* to it and can be applied to various other applications/problems.

[6]also other user, not familiar with the data source, could use it

References

[1] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.

[2] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The physiology of the grid: An open grid services architecture for distributed systems integration," July 2002. [Online]. Available: http://www.globus.org/research/papers.html#OGSA

[3] G. Stix, "The triumph of the light," *Scientific American*, 2001.

[4] D. Pearson, "Data requirements for the grid," GGF, Tech. Rep., 2003.

[5] GGF Database Access and Integration Services Working Group, http://www.cs.man.ac.uk/grid-db/.

[6] Open Grid Services Architecture Data Access and Integration (OGSA-DAI), http://www.ogsadai.org.

[7] G. Wiederhold, "Mediators in the architecture of future information systems," The IEEE Computer Magazine, March 1992.

[8] A. Wöhrer and P. Brezany, "Mediators in the Architecture of Grid Information Systems," Institute for Software Science, Tech. Rep., February 2004.

[9] University of Vienna – Institute for Software Science, "The Knowledge Grid project," http://www.gridminer.org.

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

[11] M. Kay, "SAXON: the XSLT and XQuery processor," http://saxon.sourceforge.net/.

[12] OGSA-DAI, "OGSA distributed query processing," http://www.ogsadai.org/dqp/.

[13] A. P. Sheth and J. A. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases," *ACM Computing Surveys*, vol. 22, no. 3, pp. 183–236, 1990.

[14] V. Josifovski, "Design, implementation and evaluation of a distributed mediator system for data integration," Ph. D. Thesis, The University of Linköping, Sweden, 1999.

[15] R. Yerneni, "Mediated query processing over autonomous data sources," Ph.D. dissertation, Stanford University, August 2001. [Online]. Available: citeseer.nj.nec.com/594425.html

[16] C. Baru, A. Gupta, B. Ludäscher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu, "XML-based information mediation with MIX," in *ACM-SIGMOD*, 1999, pp. 597–599. [Online]. Available: citeseer.nj.nec.com/baru99xmlbased.html

[17] K. Lee, J. Min, and K. Park, "A Design and Implementation of XML-based mediation Framework (XMF) for Integration of Internet Information Resources," in *Proceedings of the 35th Hawaii International Conference on System Sciences - 2002*, 2002.

[18] D. Kossmann, "The state of the art in distributed query processing," *ACM Computing Surveys (CSUR)*, vol. 32, no. 4, 2000.

[19] A. Gounaris, N. W. Paton, A. Fernandes, and R. Sakellariou, "Adaptive query processing: A survey," 2002. [Online]. Available: citeseer.nj.nec.com/gounaris02adaptive.html

[20] H. Wache, T. Voegele, T. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Huebner, "Ontology-based integration of information - a survey of existing approaches," 2001. [Online]. Available: citeseer.nj.nec.com/article/wache01ontologybased.html