International Journal of Computer Science and Network (IJCSN)
Volume 1, Issue 3, June 2012 www.ijcsn.org ISSN 2277-5420

Integrating Heterogeneous Data Sources Using XML Mediator

[1] Yogesh R.Rochlani , [2] Prof. A.R. Itkikar

[1]Department of Computer Science & Engineering Sipna COET, SGBAU, Amravati (MH), India

[2]Department of Computer Science & Engineering Sipna COET, SGBAU, Amravati (MH), India

Abstract

In the past decade, research works in heterogeneous database integration have established a good and solid framework to alleviate this task. However, there are still works need to be accomplished to bring these achievements to be easily implemented and integrated to Internet applications. This paper presents the XML mediator, a tool for integrating and querying disparate heterogeneous information as unified XML views. It describes the mediator architecture and focuses on the distributed query processing technology implemented in this component.

***Keywords*:** Heterogeneous, Integration, GAV, LAV, Rewriting Queries, Wrapper, XML, XQuery

.

1.Introduction

In recent years, there have been many research projects focusing on heterogeneous information integration system. The goal of such a system is to intercept the user queries and to find the more adequate data and services from several heterogeneous resources, to answer the queries of the user, to pass the specific parameters, to call upon the service and to turn over the result in a transparent way to the users. The latter do not need to know the nature, the type or the localization of the data, where the services are called upon, in which language they were programmed and on which operating system they are lodged, or no other system aspects which do not form part of the interface of the required services. Typical information integration systems have adopted wrapper mediator architecture [1]. In this architecture, mediators provide a uniform user interface to query integrated views of heterogeneous information sources. Wrappers provide local views of data sources in a uniform data model. The local views can be queried in a limited way according to wrapper capabilities.

The advantages of XML as an exchange model, (i.e., it is rich, clear, extensible and secure), makes it the best candidate for supporting the integrated data model. In addition, using XML views for local data sources hides local specificities of each system. Furthermore, the richness of the XML schema model simplifies wrapper mappings. Also, the emergence of XQuery as a powerful universal query language for XML makes it possible to query XML global and local views in a uniform way based on a standard interface. This paper gives an overview of the XML Mediator. The fourth section focuses on the middleware objectives. Then, we briefly describe the system architecture. Further we give an overview of the query processing technology embedded in the component. Further, we focus on typical applications of the mediator.

2. Related works

Data integration has received significant attention since the early days of databases. In the recent years, there have been several works focusing on heterogeneous information integration. Most of them are based on common mediator architecture [12]. In this architecture, mediators provide a uniform user interface to views of heterogeneous data sources. They resolve queries over global concepts into sub queries over data sources. Mainly, they can be classified into structural approaches and semantic approaches.

In structural approaches, local data sources are assumed as crucial. The integration is done by providing or automatically generating a global unified schema that characterizes the underlying data sources. On the other hand, in *semantic approaches*, integration is obtained by sharing a common ontology among the data sources. According to the mapping direction, the approaches are classified into two categories: global-as-view and local-as-view [13]. In *global-as-view* approaches, each item in the global schema is defined as a view over the source schemas. In *local-as-view* approaches, each item in each source schema is defined as a view over the global schema. The local-as-view approach better supports a dynamic environment, where data sources can be added

to the data integration system without the need to restructure the global schema.

There are several well-known research projects and prototypes such as Garlic [2], Tsimmis [3], MedMaker [9], and Mix [10] are structural approaches and take a global-as-view approach. A common data model is used, e.g., OEM (Object Exchange Model) in Tsimmis and MedMaker. Mix uses XML as the data model; an XML query language XMAS was developed and used as the view definition language there. DDXMI (for Distributed Database XML Metadata Interface) builds on XML Metadata Interchange. DDXMI is a master file including database information, XML path information (a path for each node starting from the root), and semantic information about XML elements and attributes. A system prototype has been built that generates a tool to do the metadata integration, producing a master DDXMI file, which is then used to generate queries to local databases from master queries. In this approach local sources were designed according to DTD definitions. Therefore, the integration process is started from the DTD parsing that is associated to each source.
Many efforts are being made to develop semantic approaches, based on RDF (Resource Description Framework) and knowledge-based integration [3]. Several ontology languages have been developed for data and knowledge representation to assist data integration from a semantic perspective, such as Ontolingua [1]. F-logic [11] is employed to represent knowledge in the form of a domain map to integrate data sources at the conceptual level. An ontology based approach [5] is one from many other researches which use ontology to create a global schema.We classify our system as a structural approach and differ from the others by following the local-as-view approach. The XML Schema language is adopted in our work instead of DTD grammar language, which has limited applicability. While only simple cases of heterogeneity conflicts among elements were handled in the paper [2], this work involves more features of XML schema components; we handle more mapping cardinality cases involving attributes in which the core purpose is to provide more information about the elements.

3. Analysis of Problem

Information Systems are expected to be a completely new generation of software systems. Their main task is to operate at a global level over existing data sources. It is important to consider that these sources have certain characteristics making the integration process very difficult:

- **Heterogeneity**: The data sources are mostly developed for a special purpose. This often results in different solutions for storing information of the same real-world objects. Information can be stored in databases with different models (relational, object oriented), or be available as Web Services. It is obviously that these kinds of sources are accessed through different interfaces, protocols and languages (Syntactical Heterogeneity). Even the same data model can cause mapping conflicts due to different understandings of the real world.

- **Autonomy**: Data sources do not give up their autonomy. First of all they keep their Design autonomy. It's up to them how the contained information is stored. Furthermore they are able to decide which other systems are allowed to communicate with them. Additionally each component is independent in deciding how the incoming queries are scheduled and executed.

- **Distribution**: Sources do not always reside on the same host. It is likely that they are on different hardware platforms and operating systems and can only be accessed through certain network protocols.

4. Proposed work

4.1 Role of the Mediator

The mediator is an interface between the user and the collection of given resource and services giving him the possibility to query a homogeneous and centralized information system by providing him with an integrated global Schema. The main operations of Mediator can be defined in the following way:

a) Query Analysis: It carries out the syntactic analysis (in accordance with grammar) and semantic in accordance with the referred view or with the query Schema.

b) Query Translation: This case of use makes it possible to translate the user's query under the XML query language,

c) Optimization of the Query: It is the main role of the mediator to divide, according to global Schema, the users' query in several sub-queries supported by the sources.

d) Translation of the Result to the User's Format: To reformulate the answer to be validated in accordance with the user's query language,

e) Mediator cache manager. Manage the semantic cache of the mediator

4.2. Functional Constraints and Architecture of the Mediator

This mediator is the result of a detailed study of the advantages and disadvantages of several existing mediators. The implementation of the core is based on the management technology of the objects distributed around the two data models: the relational/object model and the XML model. Concerning the adopted approach it is a mixed approach between GAV and LAV. This last choice is justified by the simplicity of the queries' rewriting operation, through the use of GAV approach, and also by the evolution and the flexibility of the systems with the introduction of LAV approach [12].The main contribution in the core of architecture that we propose and the originality of this system is summarized in the method of the global Schema definition which is based on the processor of refinement by specialization of the domains [13]. This new technology will be presented in the following section and the new methodology for the management of the semantic cache, and in the use of the Web Service Technology to ensure the tools integration.
The generic architecture of this mediator is illustrated in Figure 1

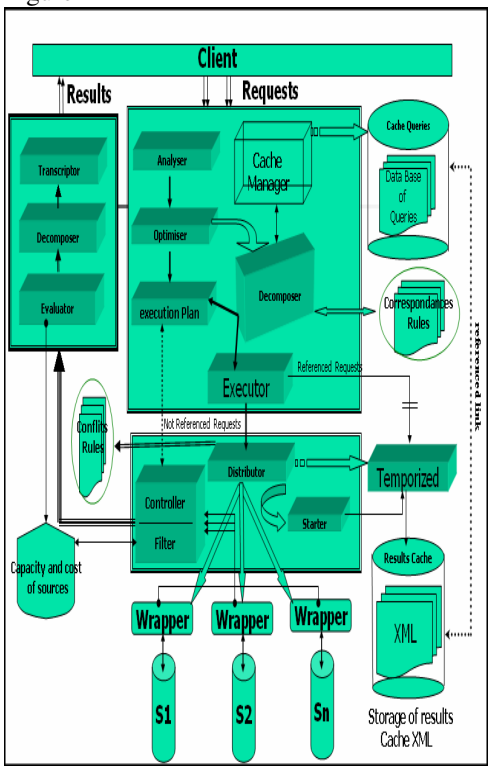

Figure 1: Mediator Architecture.

The following presents the role of the basic composites of the mediator:

Analyzer : This component allows to analyze the queries of the users for the syntactic and lexical checking.

Optimizer : This component optimizes the query according to preset rules' of optimization.

Decomposer : It carries out the operation of the query rewriting of the users. It generates the sub-queries and sends them to the specific wrappers local sources.

Execution plan Generator: It defines an execution order of deferent sub-query generated by Decomposer

Queries Executor It carries out the operation of transmission of the sub-queries to the different wrappers and to the manager of the semantic cache.

Temporization: It allows synchronizing between the execution of the sub-queries on the local sources and the semantic cache query.

Starter: It make possible to start the operations of the overlapped sub-queries execution and the data filtering.

Controller/filter : It carries out the operations of the overlapped sub-queries execution and the data filtering.

Evaluator : Control the cost of various resources.

Decomposer: It allows to combine the results received from various queried local sources and those of the semantic cache.

Transcriptor: Supplies the final result at the users

Cache Queries Database : This source contains the users queries history for the queries submitted to the mediator

Cache results Database: This source contains the users' queries execution results.

Correspondences Rules: These rules used to bind the elements of the sources Schema to those of the Global Schema (inter-Schema s correspondence)

Conflicts Rules: These rules used to manage the Mapping phase, to solve the inter-Schema conflicts and to establish the inter-Schema correspondences

Wrapper : It is responsible for wrapping a data source in such a way that the source can interact with the rest of the integration system

4.3 Global Schema and Query Processing.

In this part, we are interested in the definition of the mediator's global Schema and the necessary stages to

process the users queries and to generate sub-queries to adapt them to the different sources integrated by the mediator [9]-[11].

4.3.1. Definition of the Global Schema.

The Global Schema definition is based essentially on the identification of all the domains which model the whole of the data and services case study. These domains are modeled by a hierarchical structure, where each node represents a domain grouping subdomains defined by the children of this node. Consequently, each node of the Global Schema integration tree is characterized by: a name and a description of the domain, a list of its attributes, a list of the integrated sources, a list of the integrated tools and a list of sub-domains generated by the father domain. The main reports of the Global Schema integration definition is based on the process of successive refinement by specialization starting from a basic federator domain (i.e. Root of the integrating tree). Moreover we suppose that each source represents a view on a sub-domain in the integration tree hierarchical structure (L.A.V Approach). This Global Schema can be described by the following integration tree:

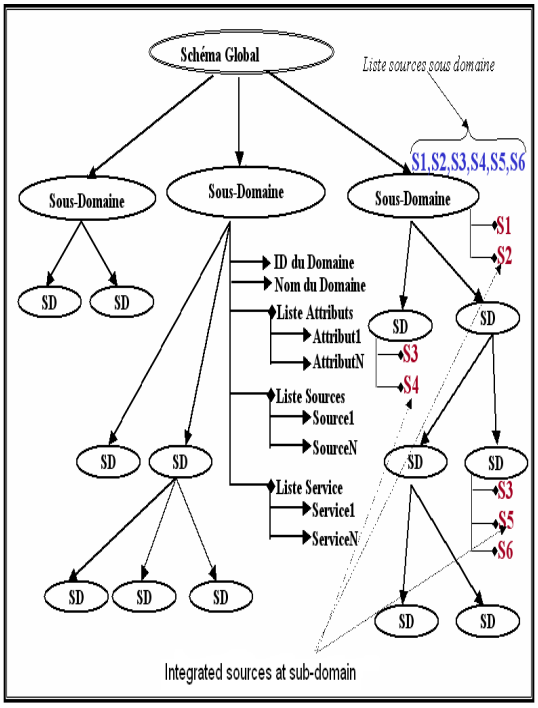

Figure 2: Integration Tree structure.

The improvement of these domains structuring allows to facilitate and optimize, at the same time, the phase of the mediator's query by the users and to generate the queries execution plan. Indeed, the user can easily explore the integration Global Schema tree to determine an optimum list of sources which can be queried by the mediator. Indeed, after having to carry out the rewriting of requests and affecting each sub-request a specific Domain in the tree of Global Schema, the mediator generates a plan of execution preestablishes, following an in-depth course of the tree. The results of execution of each sub-request are stored in the temporary memories associated the domain of the tree. At the time of the customer request evaluation and to generate the finale result required by the customer, these data will be amalgamated starting from the answers partial recorded to the level of the sheets to the root of the tree of Global Schema. Consequently, in the mapping phase only the necessary sources are treated. This structure will also enable us to define an execution plan of sub-queries generated by the mediator. After the rewriting phase of a query, the order of execution starts with subqueries generated for the sources integrated into the low level domains (possibly sheets) until the federator domain For the Global Schema integration definition the following basic constraints have been proposed:

1. A source can be integrated by several domains.
2. The list of the sources integrated by a domain is the list of all the sources integrated by all sub domains of this domain.
3. Sub-domains are disjoined: sources can be affected only to one and one sub-domain of the same domain.
4. If two domains (or several) of the same levels (even depth in the integration tree) integrate same sources, the level of integration of this sources moves on the level of the father domain of these domains.

4.3.2 Definition of the Mapping Schema

One of the main problems arising from the data integration consists in carrying out the correspondence between a data source schema and the Global Schema . Generally, it is a question of laying down the rules which make it possible to bind the elements of the Schema of a source to those of the Global Schema (inter-Schema s correspondence). This makes it possible to the mediator to answer the queries of the user which are submitted on the Global Schema .

Correspondences Identification

When the Global Schema reaches the desired level of conformity, the following stage consists in identifying the common correspondence rules. With each time that is possible, the correspondences are defined in intention. The integration process consists in finding these correspondences between the elements of the sources and

those of the Global Schema. These rules form part of the integration process result. In our model, the elements in correspondences can be entities, attributes or many access paths to the attributes and methods signature, etc. These elements are varied according to the sources model (i.e. Relational, object, XML.). The correspondence elements can be summarized in the following table

	source Element	global Schema Element
Relational	Relation	Entity
	Attribute	Attribute
	Trigger	Service
Object	Class	Entity
	Attribute	Attribute
	Method	Service
XML	Entity	Entity
	Attribute	Attribute
	Path	Path

Table 1: The elements schema correspondence

In order to well manage the mapping phase and to solve the inter-Schema conflicts problems; a list of basic rules has been defined to establish the following inter-Schema correspondences:

1. Each source can be identified by a view on a part of the integration tree of the Global Schema (GAV approach).
2. If a source element is in correspondence between two elements of two different domains, the constraints on the choice of a correspondence must be fixed for the management of the inter-Schema conflicts.
3. If two elements of the same source are in correspondence with two elements of two different domains, the priorities between these two domains must be defined for the generation of the execution plan.
4. Each element of a source can be in correspondence only with one element of the source integration domain sub-domains.

Query Processing

Firstly, Mediator receives a query formulated in terms of the unified Schema and queries the cache manager, which generates tow sub-queries: the local query and the distance query. The first query is used to extract the local data stored in semantic cache. The second query is decomposed by the rewriter component into sub-queries and addressed to specific data sources. This decomposition is based on source descriptions by global Schema and mapping Schema , which play an important role in sub-queries' execution plan optimization. Finally, the sub-queries are sent to the wrappers of the individual sources, which transform them into queries over the sources. The results of these subqueries are sent back to the mediator. At this point the answers are merged with the result of cache query by the local query and returned to the user. Besides the possibility of making queries, the mediator has no control over the individual sources.

The latter component (wrapper) is responsible for wrapping a data source in such a way that the source can interact with the rest of the integration system. It provides the mediator with data from the source that it is in charge of, as asked by the query execution engine. In consequence, it presents a data source as a convenient database, with the right Schema and data, appropriate for being understood and used by the mediator. This presentation Schema may be different from the real one, i.e., the internal to the data source.

4.4. Example

Here is an example that shows two data bases with different semantic conflicts.

There are two databases containing semantically related information about books but in different formats. Sites X and Y contain tables named products and productslist respectively. There are some semantic discrepancies between these sites. They are listed as follows:

1. **There are attribute-to-attribute conflicts:** The attributes products.pno and Products.name in Site X are respectively named productlist.pid and productlist.productname in Site Y.
2. **There is value-to-value conflict:** The productlist.location stores more detailed data than products.location.
3. **There is a table-to-table conflict:** The products.component is missing in the relation productlist. Besides, the productlist.manufacturing_year is also missing in the relation products.

Products (Site X)

pno	name	company	cost	dealer	location	components
1	MotherBoard	Heiss	2000	IBM Company	India	100
2	Micro Controllers	Joe	2500	Microns Company	USA	65

Productslist(Site Y)

pid	productname	company	cost	dealer	location	Manufacture_year
1	MotherBoard	Heiss	2000	IBM	Coimbatore,India	2000
2	Micro Controllers	Joe	2500	Microns	Clevelan d,USA	2009

Figure 3. Two Databases with different schemas.

The two tables products and productslists can be integrated as ProductsData(pno, name, company, cost, dealer, location, components, manufacture_year). When the user wishes to show ProductsData.cost by the original list price, ProductsData.dealer by the full names, and ProductsData.location by the concise names two sets of XSLT and template files are used to transform both tables into ProductData, respectively. Figure 4 and Figure 5 show the XSLT and template files for Site X. For Site Y, the XSLT and template files are shown in Figure 6 and Figure 7 respectively. Finally, both tables can be outer-joined into ProductData as Table 2. illustrates.

```
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Tran
sform" version="1.0">
<xsl:template match = '*'>
<xsl:apply-templates />
</xsl:template>
<xsl:template match = 'products'>
<TR><TD><xsl:value-of select = '@pno'
/></TD>
<TD><B><xsl:value-of select = '@name'
/></B></TD>
<TD><B><xsl:value-of select = '@company'
/></B></TD>
<TD><B><xsl:value-of select = '@cost'
/></B></TD>
<TD><B><xsl:value-of select = '@dealer'
/></B></TD>
<TD><B><xsl:value-of select = '@location'
/></B></TD>
```

Figure 4(a) The XSLT for site X

```
<TD><B><xsl:value-of select =
'@components' /></B></TD>
</TR>
</xsl:template>
<xsl:template match = '/'>
<HTML>
<HEAD>
<STYLE>th { background-color: #CCCCCC
}</STYLE>
</HEAD>
<BODY>
<TABLE border='1' style='width:600;'>
<TR><TH colspan='7'> ProductsData
</TH></TR>
<TR><TH>pno</TH><TH>Name</TH><TH>compnay</
TH><TH>cost</TH>
<TH>dealer</TH><TH>location</TH><TH>compon
ents</TH></TR>
<xsl:apply-templates select = 'ROOT' />
</TABLE>
</BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>
```

Figure 4(b) The XSLT for site X

```
<ROOT xmlns:sql="urn:schemas-microsoft-
com:xml-sql" sql:xsl="Products.xsl">
<sql:query>
SELECT pid, name, company, cost, dealer,
location, components FROM products
FOR XML AUTO
</sql:query>
</ROOT>
```

Figure 5 Template files for Site X

```
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Tran
sform" version="1.0">
<xsl:template match = '*'>
<xsl:apply-templates />
</xsl:template>
<xsl:template match = 'productlist'>
<TR>
<TD><xsl:value-of select = '@bid' /></TD>
<TD><B><xsl:value-of select =
'@productname' /></B></TD>
<TD><B><xsl:value-of select = '@company'
/></B></TD>
<TD><B><xsl:value-of select = '@cost'
/></B></TD>
<TD><B><xsl:value-of select = '@dealer'
/></B></TD>
<TD><B><xsl:value-of select = '@location'
/></B></TD>
```

Figure 6. The XSLT for site Y

```
<ROOT xmlns:sql="urn:schemas-microsoft-
com:xml-sql" sql:xsl="productlist.xsl">
<sql:query>
SELECT pid, productname, dealer, cost/0.4
as cost,
rtrim(publisher) + ' Company' as dealer,
right(rtrim(location), 2) as location,
manuafactureyear
FROM productlist
FOR XML AUTO
</sql:query>
</ROOT>
```

Figure 7. Template files for Site Y

pno	Name	company	Cost	dealer	location	components	manufacture_year
1	MotherBoard	Heiss	2000	IBM Comp	India	100	2000

				any			
2	MicroController	Joe	2500	Microns Company	USA	65	2009

Table 2. Integrated Relation ProductData in the Global site.

5. Applications

Several applications are currently built using the e-XML Mediator. A simple kind of application is the publishing of relational data as integrated data in XML. We packaged our XML/DBC wrapper for object-relational databases as a component marketed under the name XMLizer, to transform any relational source in an XML data source supporting XQuery. For example, the XMLizer is a key component in several database interchange, XML EDI and XML portal applications. More complex applications using the Mediator include portals for querying multiple heterogeneous databases. We have also developed applications in cooperation with European partners for a tourism Web site federating multiple data sources, for a virtual hospital federating patient dossiers constituted from several pieces, and for an active document publisher composing documents from several sources including databases and reports. In general, the mediator is ideal for extracting and composing disparate information as unique XML documents. Coupled with the other products of e-XMLMedia, XML Repository and XForms Engine (XFE), the Mediator and XMLizer are ideal to develop gateways between existing information systems and new XML consuming applications.

6. Conclusion

The proposed architecture satisfies almost all requirements for a mediator allowing an efficient integration of heterogeneous information systems. Besides the integration of different kinds of data sources it offers now a more flexible way of extending the system. Our Mediation system currently provides following features:

- With such mediation architecture for information systems it is possible to make several information systems with different designs and architectures cooperate.
- Several heterogeneous data sources can be easily integrated, updated or just removed from the system by simply changing the global Schema.
- A large amount of available databases, structured text files and Web Services are supported due to already available wrappers. Of course it is possible to write own wrappers that import other currently not supported data sources.
- The mapping process is carried out by certain simple mapping actions.

References

[1] Wiederhold G.: "Intelligent Integration of Information", ACM SIGMOD Conf. On Management of data, pp. 434-437, Washington D.C., USA, May 1993.

[2] Haas L., Kossman D., Wimmers E., Yang J.: "Optimizing Queries across Diverse Data Sources", 23rd Very Large Data Bases, August 1998, Athens, Greece, 1997.

[3] Chawathe S., Garcia-Molina H., Hammer J., Ireland K., Papakonstantinou Y., Ullman J., and Widom J.: "The TSIMMIS Project : Integration of Heterogeneous Information Sources", IPSJ Conference, pp. 7-18, Tokyo, Japan, October 1994.

[4] Fankhauser P., Gardarin G., Lopez M., Muñoz J., Tomasic A.: "Experiences in Federated Databases: From IRO-DB to MIRO-Web", 24rd Very Large Data Bases, pp. 655-658, August 24-27, 1998, New York City, New York, USA, 1998

[5] Cluet S., Delobel C., Siméon J., Smaga K.: "Your Mediators Need Data Conversion", ACM SIGMOD Intl. Conf. on Management of Data, pp. 177-188, Seattle, Washington, USA, 1998.

[6] Christophides V., Cluet S., Siméon J.: "On Wrapping Query Languages and Efficient XML Integration", ACM SIGMOD 2000, pp. 141-152, May 16-18, 2000, Dallas, Texas, USA. SIGMOD Record 29(2) ACM 2000.

[7] Manolescu I., Florescu D., Kossmann D.: "Answering XML Queries over Heterogeneous Data Sources", 27th Very Large Data Bases, pp. 241-250, Roma, Italy, Sept. 2001.

[8] Shanmugasundaram J., Kiernan J., Shekita E., Fan C., Funderburk J.: "Querying XML Views of Relational Data",Proc. Of the 27th International Conference on Very Large Data Bases, pp. 261-270, Roma, Ital., Sept. 2001.

[9] PAPAKONSTANTINOU, Y., GARCIA-MOLINA, H., ULLMAN, J., MedMaker: A Mediation System Based on Declarative Specifications, in: Proc. of the IEEE Int.

Conf. on Data Engineering, New Orleans, LA, February 1996, pp. 132-141.

[10] BARU, C., GUPTA, A., LUDASCHER, B, MARCIANO, R., PAPAKONSTANTINU, Y., VELIKHOV, P., and CHU, V., XML-Based Information Mediation with MIX, in: Proc. of the ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 597-599.

[11] NAM Y., GOGUEN, J., WANG, G., A Metadata Integration Assistant Generator for Heterogeneous Distributed Databases, in: Proc. of the Confederated International Conferences DOA, CoopIS and ODBASE, Irvine CA, October 2002, LNCS 2519, Springer, pp. 1332-1344.

[12] WIEDERHOLD, G., Mediators in the Architecture of Future Information System, in: IEEE Computer Magazine, Vol. 25, No. 3, March 1992, pp. 38-49.

[13] LENZERINI, M., Data Integration: A Theoretical Perspective, in: Proc. of the ACM Symposium on Principles of Database Systems, Madison, Wisconsin, USA, June 2002, pp. 233-246.

[14] MAY, W., A Rule-Based Querying and Updating Language for XML, in: Proc. of the Workshop on Databases and Programming Languages, Springer LNCS 2397, 2001, pp. 165-181.

[15] S. Cluet, C. Delobel, J. Siméon, K. Smaga, "*Your* Mediators Need Data Conversion", ACM SIGMOD Intl. Conf. on Management of Data, pp. 177-188, Seattle, Washington, USA, 1998.