Using Agents for Generation and Maintenance of Mediators

Bernadette Farias Lóscio
Centro de Informática
Universidade Federal de Pernambuco
bfl@cin.ufpe.br

Ana Carolina Salgado
Centro de Informática
Universidade Federal de Pernambuco
acs@cin.ufpe.br

Vânia Maria Ponte Vidal
Departamento de Computação
Universidade Federal do Ceará
vvidal@lia.ufc.br

Abstract *In this paper we present a system for data integration on the web, where an XML-based mediator plays a key role providing a homogeneous view of different data sources. One novelty of our approach is that we also propose solutions for the problems of generation and maintenance of mediators. Observe that, in dynamic environments, such as the Web, individual data sources may change not only their data but also their schemas. As a result, whenever a local schema changes, the mediator needs to be updated to reflect the modifications. The proposed system uses agents to support mediator generation and maintenance. We specify a set of tasks that must be performed in order to support both generation and maintenance of mediators. In our approach, we use correspondence assertions for specifying the semantics of XML-based mediators.*

Keywords: *Mediation system, mediation queries, data integration, XML, maintenance of mediators*.

1 Introduction

Several systems [1, 2, 3, 4, 8, 9] have been built with the goal of integrating data from multiple web sources. Many data integration systems use the mediator architecture [22] to provide integrated access to multiple data sources that can be autonomous and heterogeneous. A mediator supports a mediation schema and a set of mappings between the mediation schema and the local data sources. These mappings, called mediation queries, are used to compute the objects in the mediation schema when queries are posed to the mediator. Queries submitted to the mediator are decomposed at run time into queries on the local data sources. The results from these queries on the local data sources are translated, filtered and merged, and then the final answer is returned either to the user or to the application.

Mediation systems can be classified according to the approach used to define the mediation mappings between the data sources and the global schema [21, 13, 15]. The first approach is called global-as-view (GAV) and requires that each object of the global schema be expressed as a view (i.e a query) on the data sources. Several projects, like Tsimmis [8], YAT [9] and Disco [18] adopt the GAV approach. In the other approach, called local-as-view (LAV), mediation mappings are defined in an opposite way; each object in a given source is defined as a view on the global schema. An example of system which uses the LAV approach is the Information Manifold system [14].

In this work, we present an agent-based system for data integration on the Web. A distinguishing feature of our system is that besides integrating data it also deals with the problems concerning generation and maintenance of mediation queries. The proposed system adopts the GAV approach to define the mappings between the data sources and the mediation schema.

As we know, one of the difficulties in integrating information from multiple data sources is the heterogeneous structure of them. To overcome this limitation integration systems use a common data model for representing the sources' content and structure. We use XML [5] as a common data model for data exchange and integration. In XML-based information integration systems, mediation queries are defined in a declarative language specifically designed for XML [7, 10]. Specifying mediation queries usually requires a fair amount of knowledge about the concepts in the underlying data sources and about the correspondences between those concepts and the ones in the mediator schema. As presented in [20], we use correspondence assertions to formally specify the relationship between the mediation schema and the data sources schemas. An advantage of using correspondence assertions is that the mediation queries generation can be automated.

Most of the approaches used in information integration systems are based on previously defined static views which gather information from a fixed set of heterogeneous data sources and provide the user with an uniform view of the distributed information. Their main limitation is related to the capability of evolving according to dynamic information systems. In this paper, we address issues related to the evolution and maintenance of XML-based information integration systems. Such issues include evolving mediation queries under schema-level changes of data sources to reflect the modifications occurring in data source schemas.

The remainder of this paper is organized as follows. In section 2 we present an architectural overview of our system and our key design decisions. In section 3 we describe how agents are used for generation and maintenance of mediation queries. In section 4 we present some related works. Finally, in section 5 we present our conclusions and suggestions for further research.

2 Architectural Overview

We propose a mediation-based data integration system that offers an integrated view of several heterogeneous web data sources. This system presents solutions for the problems concerning mediation queries generation and maintenance in dynamic environments. As shown in Figure 1, the system architecture can be divided into three spaces:

- Common core: this space feeds the mediator generation and maintenance space with information about local data source schemas while receiving local data source queries from the data integration space and answering them.
- Data integration space: this space is composed by the mediator responsible for restructuring and merging data from autonomous sources and for providing an XML integrated view of the data.
- Mediator generation and maintenance space: through semi-automated processes this space executes the mediation queries generation and maintenance. The mediation queries generation process consists of three steps:
  - *Mediation schema modeling*: this step analyzes the user requirements and specifies the mediation schema using a high-level data model.
  - *Mediation schema integration*: this step integrates the mediation schema with the local schemas in order to identify the correspondence assertions that formally specify the relationships between the mediation schema and the local schemas. To accomplish this, the mediation schema and the local schemas should all be expressed in the same data model, the so-called “common” data model. Therefore, a translation of the local schemas to the common data model is necessary.
  - *Mediation queries generation*: this step generates the mediation queries based on the mediation schema and the mediator’s correspondence assertions. Correspondence assertions are special types of integrity constraints which are used to assert the correspondences among schemas.

Besides the mediation queries generation, this space is also responsible for the maintenance of the mediation queries under schema-level changes of data sources.

In what follows, we present an overview of our system based on these spaces. We also present our key design decisions, including the common data model used to represent the local schemas and the mediation schema, the formalism to capture the correspondences among schemas and the mediation queries definition language.

2.1 Common Core

This space is related to the two other spaces and is composed by the following components: the data sources, the wrappers and the middleware. In the following these components are presented in more detail.

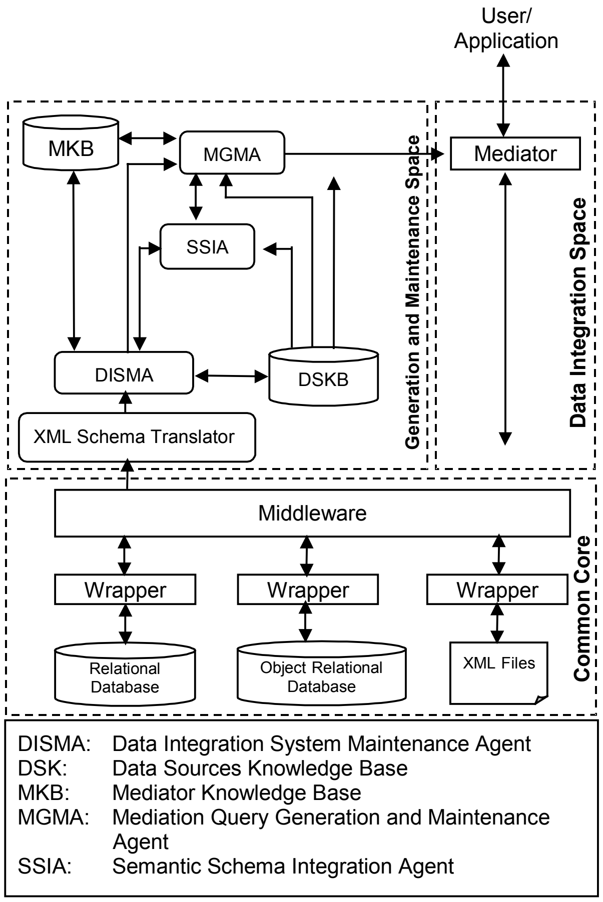
User/
Application
MKB
MGMA
Mediator
SSIA
DISMA
DSKB
XML Schema Translator
Generation and Maintenance Space
Data Integration Space
Middleware
Wrapper
Wrapper
Wrapper
Relational Database
Object Relational Database
XML Files
Common Core
DISMA: Data Integration System Maintenance Agent
DSK: Data Sources Knowledge Base
MKB: Mediator Knowledge Base
MGMA: Mediation Query Generation and Maintenance Agent
SSIA: Semantic Schema Integration Agent

Figure 1: Architectural Overview

- *Data sources*

The data sources can be heterogeneous, autonomous and often dynamic. This is due to the fact that the data sources support local applications and update their data and schemas independently, possibly without any concern of how this may affect the data integration system based upon them. Data sources may also be added to the system, or become temporally or definitively unavailable. A data source is included in the integration system via a wrapper that serves as a bridge between the data source and the other components of the system. When a data source joins the system, it publishes its exported schema describing the information available through this data source. It is important to note that when a data source changes its exported schema it is necessary to publish it again. The exported schema must be updated to reflect the corresponding local schema changes or when some information needs to be added or dropped from it. The data sources ideally publish the most recent version of their exported schema in order to keep the consistency between the information available to the user and the data actually stored in the data source.

- *Wrappers*

Wrappers are necessary for each data source to translate application queries into source specific queries and to translate the data returned by the local data sources into the common data model. In this work, we use XML as a common data model for data exchange and integration. Due to the flexibility of XML to represent both structured and semi-structured information there is an increasing interest in using XML as a common data model for data exchange and integration.

It is important to observe that wrappers are responsible only for the translation of data and queries, i.e., wrappers are not responsible for the translation of schemas to a common data model nor for the extraction of metadata on local data sources. In order to perform the translation of exported schemas to a common data model we use the XML Schema Translator described in section 2.3.1. Some research projects propose solutions to the problem of building wrappers for translating relational data to XML data [12, 19].

- *Middleware*

The middleware interacts with the data integration space receiving a set of queries from the mediator and forwarding them to the wrapper of the corresponding data sources. When a wrapper receives the results for these queries they send them to the middleware which returns them to the mediator.

The middleware also interacts with the mediator generation and maintenance space sending the exported schemas published by the local data sources to the XML Schema Translator.

2.2 Data Integration Space

This space is composed by the mediator responsible for the activities involving integration of data distributed in several web data sources. In what follows, we describe the mediator in more details.

- *Mediator*

A mediator is a software device that supports a mediation schema which captures the user requirements, and a set of mappings, called mediation queries, between the mediation schema and the local data sources. Mediation queries are used to compute the objects in the mediation schema when queries are posed to the mediator. As presented in Figure 2, each concept C_i in the mediation schema is associated to a mediation query Q_i which computes the concept C_i over the set of data sources.

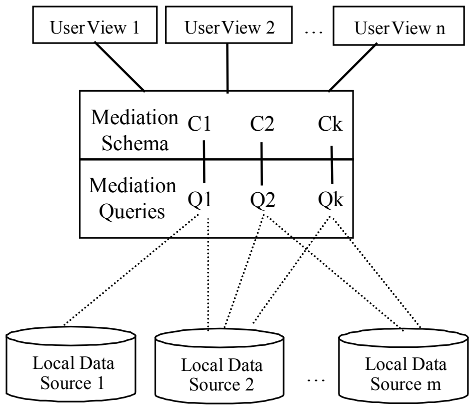

Figure 2: Mediator Description

2.3 Mediator generation and maintenance space

The main goals of this space are the generation and maintenance of the mediation queries. The components of this space are: i) *XML Schema Translator*; ii) *Semantic Schema Integration Agent (SSIA)*; iii) *Mediator Knowledge Base (MKB)*; iv) *Data Sources Knowledge Base (DSKB)*; v) *Mediation Queries Generation and Maintenance Agent (MGMA) and vi) Data Integration System Maintenance Agent (DISMA)*. In the following we present these components.

- *XML Schema Translator*

When the local data sources publish their exported schemas defined in their own data model (ex: relational or object-oriented) it is necessary to translate them to a common data model in order to perform the schema integration. The component responsible for this translation is the XML Schema Translator. We use XML Schema [6] to represent schemas from multiple web data sources and to represent the mediation schema.

We also use a diagram to illustrate the structural information of XML schemas. Thus, it is possible to have a better understanding of the semantics associated with the local schemas. Consider, for example, the XML Schema presented in Figure 3, which contains information about students and courses from a Computer Science Department of a university. Figure 4 illustrates the tree-structured representation for the Computer Science Department schema. A tree model consists of a set of trees, one tree for each type specified in the XML schema. Note that (i) bold fonts denote a type identifier and (ii) the * symbol denotes multiple occurrences of an element. In addition, leaves may also be labeled with types (e.g. *Course*).

- *Semantic Schema Integration Agent (SSIA)*

This agent has two main tasks: i) the integration of the exported schemas in order to identify the global correspondence assertions (GCA) which formally specify the relationship among the exported schemas and ii) the integration of the exported schemas with the mediation schema in order to identify the mediator correspondence assertions (MCA) which formally specify the relationship between the exported schemas and the mediation schema. It is important to note that the SSIA must interact with the mediator builder in order to solve conflicts that may appear during the schema integration process [11].

GCAs facilitate the incremental integration of the mediation schema with the exported schemas and are very important in finding appropriate replacements for mediation queries components when the mediation query becomes undefined. As mentioned earlier, MCAs help the generation of the mediation queries and are also used to automate the process of mediation queries maintenance.

In [20] we demonstrated how the correspondence assertions formally specify the relationships between XML schemas. To illustrate this, take the *Computer Science Research* schema presented in Figure 5, containing data about projects and members from the Computer Science Department: integrating the *Computer Science Department* schema (Figure 4) with the *Computer Science Research* schema we obtain the following global correspondence assertions:

GCA_1: cs_department/students/student* $\cap$ cs_research/member*

GCA_2: student/name $\equiv$ member/name

GCA_3: student/phone $\equiv$ member/phone

The **GCA_1** specifies that there is an intersection between the set of students and the set of members. In our example, the *name* element (**GCA_2**) is used for matching *students* elements in *cs_department/students/student** with *members* elements in *cs_research/member**. A *student* s_1 in *cs_department/students/student** matches a *member* m_1 in *cs_research/member** iff s_1/*name*/data( ) = m_1/name/data( ).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
  <xsd:element name=" cs_department" type= " CS_Department "/>
  <xsd:complexType name= " CS_Department ">
    <xsd:element name="courses" type="Courses"/>
    <xsd:element name="students" type=" Students "/>
  </xsd:complexType>
  <xsd:complexType name="Courses">
    <xsd:element name="course" type="Course" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:complexType>
  <xsd:complexType name="Students">
    <xsd:element name="student" type="Student" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:complexType>
  <xsd:complexType name="Course">
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="courseNo" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="Student">
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="phone" type="xsd:string"/>
    <xsd:element name="Courses" type="Courses"/>
</xsd:complexType>
<xsd:schema />
```

Figure 3: XML Schema for the Computer Science Department

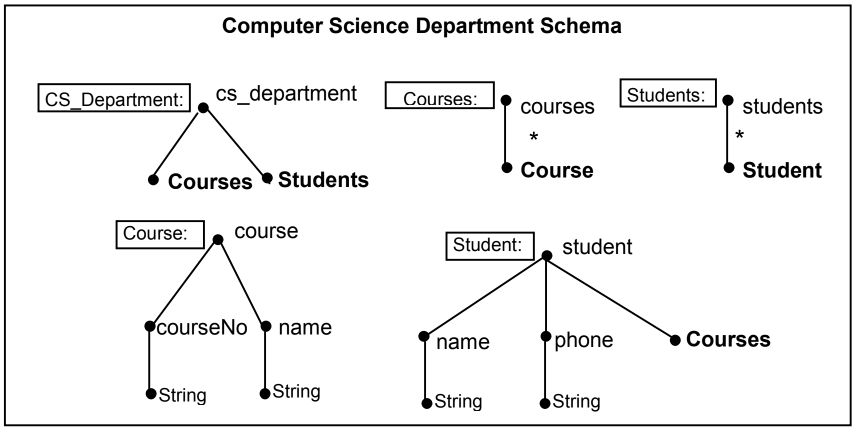

Figure 4: Schema for the Computer Science Department

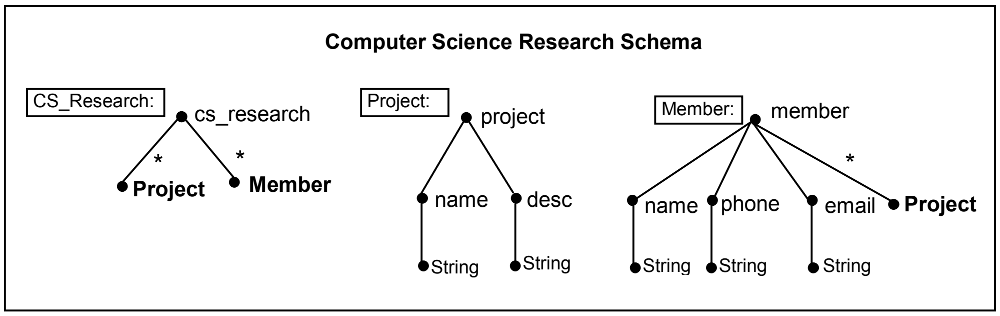

Figure 5: Schema for the Computer Science Research

- *Data Sources Knowledge Base (DSKB)*

The Data Sources Knowledge Base stores data source descriptions, including the exported schemas defined in XML Schema and the global correspondence assertions. The data source descriptions collected in the DSKB are very important in finding appropriate replacements for mediation queries components when the mediation query becomes undefined. The data sources descriptions are also important for the mediation queries generation and for translating application queries into precise query plans.

- *Mediator Knowledge Base (MKB)*

The Mediator Knowledge Base stores mediator descriptions, including the mediation schema defined in XML Schema and the mediator correspondence assertions. As discussed in section 3, the mediator description is very important for the process of mediation queries maintenance.

- *Mediation queries Generation and Maintenance Agent (MGMA)*

This agent provides a Graphical User Interface (GUI) for defining mediation schemas in a diagrammatic way. This interface displays a menu of registered data sources where each entry shows one of the data source descriptions stored in the DSKB. The mediator builder is responsible for the mediation schema definition.

This agent also interacts with the SSIA sending it the mediation schema while receiving the mediator correspondence assertions that formally specify the relationships between the mediation schema and the exported schemas stored in the DSKB. Using the mediation schema and the mediator correspondence assertions this agent generates the mediation queries. We use XQuery [7] for the definition of the mediation queries. XQuery, as proposed by the W3C, is a query language specifically designed for XML and allows XML data to be queried, translated and integrated.

Other important tasks of this agent are related to the mediation queries maintenance. When the MGMA receives an update notification it determines the appropriate actions required to update the mediation queries which consists in finding valid replacements for the mediation queries components that become invalid after a source schema change.

- *Data Integration System Maintenance Agent (DISMA)*

This agent is responsible for the maintenance of the following components: i) *Data Sources Knowledge Base*: it consists of keeping the data sources descriptions consistent with the schemas of the local data sources, ii) *Mediator Knowledge* Base: it consists of keeping mediator correspondence assertions and the mediation schema consistent with the schemas of the local data sources.

3 Using Agents for Generation and Maintenance of Mediators

In this section, we discuss more precisely how we use agents for generation and maintenance of mediation queries. First, we describe the process of mediation queries and subsequently we present the steps involved in the process of mediation queries maintenance.

- *Mediation Queries Generation*

Step 1: Mediation schema modeling

The local data sources publish their exported schemas and the XML Schema Translator translates them to XML Schema. The Semantic Schema Integration Agent (SSIA) integrates the exported schemas and identifies the set of global correspondence assertions (GCA) that formally specify the relationship among them. Based on the exported schemas and the *GCAs* the mediator builder defines the mediation schema (S_M).

Step 2: Mediation schema integration

The SSIA integrates the mediation schema with the exported schemas and identifies the set of mediator correspondence assertions ($\{MCA\}$) that formally specify the relationship between the exported schemas and the mediation schema.

Step 3: Mediation queries generation

Using the mediation schema (S_M) and the set of mediator correspondence assertions ($\{MCA\}$) the Mediation Queries Generation and Maintenance Agent (MGMA) generates the mediation queries.

- *Mediation Queries maintenance*

We propose a solution for the mediation queries maintenance problem which can be stated as follows: given a change event occurring at the source level, how to propagate this change into the mediation queries. Our solution can be decomposed in the steps presented below.

Step 1: DSKB Updating

When a new data source is included in the system and its exported schema is published and translated to XML Schema the SSIA executes the semantic integration of the new exported schema with those already stored in the Data

Sources Knowledge Base (DSKB). After the integration process, the new exported schema and the new GCAs resulting from the integration process are stored in the DSKB.

It is important to note that the DISMA can also receive an exported schema corresponding to a new version of a schema already stored in the DSKB. In this case, the DISMA compares the schema currently stored in the DSKB with its new version, in order to identify what kind of updates were applied to the original schema. To maintain the consistency of the DSKB, the DISMA determines the appropriate actions required to reflect the schema updates. For example, if the update schema drops one type of an exported schema then all global correspondence assertions referring to the dropped type must be removed from the DSKB.

Step 2: Update Notification

After updating the DSKB, the DISMA sends an update notification to the Mediation Queries Generation and Maintenance Agent (MGMA) about the exported schema update. The update notification specifies the update type and all the information required for the definition of the appropriate actions needed for updating the Mediator Knowledge Base (MKB) and the correct updating of the mediation queries. The information to be passed to the MGMA depends on the update type.

Step 3: MKB Updating

When the MGMA receives the update notification it determines the appropriate actions required to update the Mediator Knowledge Base (MKB). The updating of the MKB includes the updating of the mediator schema and the updating of the mediator correspondence assertions. Again, these actions depend on the update type. For example, when the schema update consists of the addition of a new type to an existing schema, there must be an interaction with the mediator builder in order to discover whether this new type should be added to the mediator schema. If positive, the mediator schema stored in the MKB is updated and is sent to the SSIA to identify the mediator correspondence assertions that define how the elements of this type will be synthesized from source elements.

Step 4: Mediation queries Synchronization

After the MKB updating, the MGMA executes the mediation queries synchronization, i.e., the rewriting of the mediation queries, so as to reflect the exported schema update. To guarantee this, the MGMA finds valid replacements for affected components of the existing mediation queries. The MGMA finds an acceptable view redefinition for the mediation queries based on the type of schema update and the MCAs and GCAs affected by the schema update.

3.1 An Example

In this section we present an example describing the mediator generation and mediator maintenance based on the steps described in the previous section.

- *Mediation Queries Generation*

Using the *Computer Science Department* schema (Figure 4), the *Computer Science Research* schema (Figure 5) and the global correspondence assertions specifying the relationships between the two schemas, the mediator builder defines the mediation schema *StudentMember* presented in Figure 6. This mediation schema integrates data about students and data about members of research groups from the Computer Science Department.

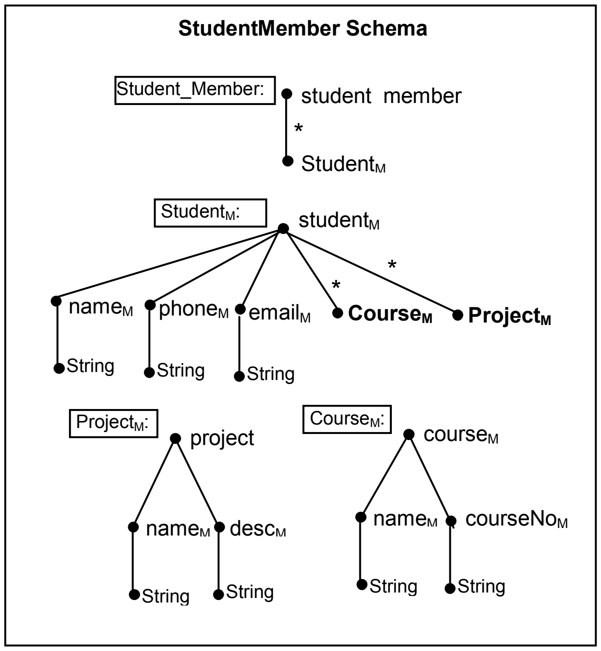

Figure 6: Schema for the Mediator StudentMember

MCA$_1$: student_member/student$_M$* $\equiv$
cs_department/students/student* $\cap$
cs_research/member*

MCA$_2$: student$_M$/name$_M$ $\equiv$ student/name

MCA$_3$: student$_M$/phone$_M$ $\equiv$ student/phone

MCA$_4$: student$_M$/course$_M$* $\equiv$ student/courses/course*

MCA$_5$: student$_M$/name$_M$ $\equiv$ member/name

MCA$_6$: student$_M$/phone$_M$ $\equiv$ member/phone

MCA$_7$: student$_M$/email$_M$ $\equiv$ member/email

MCA$_8$: student$_M$/project$_M$* $\equiv$ member/project*

MCA$_9$:course$_M$/name$_M$ $\equiv$ course/name

MCA$_{10}$:course$_M$/courseNo$_M$ $\equiv$ course/courseNo

MCA$_{11}$:project$_M$/name$_M$ $\equiv$ project/name

MCA$_{12}$:project$_M$/desc$_M$ $\equiv$ project/desc

Figure 7 : Mediator Correspondence assertions for the mediator StudentMember

```
<student_member>
  for $t in document ("csdepto.xml") //students/student <!-- from MCA1 -->
  where some $m in document ("csresearch.xml") //member/[name = $t/name] <!--from MCA1 -->
  return
    <studentM>
      <nameM> $t/name </nameM> <!-- from MCA2 -->
      <phoneM> $t/phone </phoneM> <!-- from MCA3 -->
      <emailM> $m/email </emailM> <!-- from MCA7 -->
      for $c in document ("csdepto.xml") //courses/course <!-- from MCA4 -->
      return
        <courseM>
          <nameM> $c/name </nameM> <!-- from MCA9 -->
          <courseNoM> $c/courseNo </courseNoM> <!-- from MCA10 -->
        </courseM>
      for $p in document ("csresearch.xml") //project <!-- from MCA8 -->
      return
        <projectM>
          <nameM> $p/name </nameM> <!-- from MCA11 -->
          <descM> $p/desc </descM> <!-- from MCA12 -->
        </projectM>
    </studentM>
</student_member>
```

Figure 8 : The mediation query StudentMember$_v$

When the SSIA integrates the mediation schema with the exported schemas we obtain the mediator correspondence assertions presented in Figure 7. Using the mediation schema *StudentMember* and the mediator correspondence assertions the MGMA generates the mediation query *StudentMember$_V$* (Figure 8).

The mediation query *StudentMember$_V$* has a FLWR[1] [7] expression that extracts information about *student* elements from the local source "csdepto.xml" and information about *member* elements from the local source "csresearch.xml". When an element in the local source "csdepto.xml" represents a student who is also a member of some research group then the *student* data and the *member* data are combined. As we can observe, this expression correctly

[1] A FLWR (pronounced "flower") expression is a form of XQuery expression construct from FOR, LET, WHERE, and RETURN clauses. A FLWR expression binds values to one or more variables and then uses these variables to construct a result.

implements the intersection as specified by the correspondence assertion **MCA_1**.

The information that must be returned in the RETURN-clause of the mediation query *$StudentMember_V$* is based on the correspondence assertions that specify the correspondences of the nested elements in *$student\_member/student_M$**, in *cs_department/ students/student** and in *cs_research/member**. For example, lines 6, 7 and 8 of the *$StudentMember_V$* query are based on the correspondence assertions **MCA_2**, **MCA_3** and **MCA_7** respectively. The correspondences of the nested elements in *$student\_member/student_M/course_M$** with those in *cs_department/students/student/courses/ course** are used to generate the lines 12 and 13 (from **MCA_9** and **MCA_{10}**). Analogously, the correspondences of the nested elements in *$student\_member/student_M/project_M$** with those in *cs_research/member/project** are used to generate the lines 18 and 19 (from **MCA_{11}** and **MCA_{12}**).

- *Mediator Maintenance*

Step 1: DSKB Updating

After the mediation queries generation, suppose the Computer Science Department publishes a new version of its exported schema. As presented in Figure 9, the corresponding *Student* type was modified by the removal of the element *phone*. The SSIA compares this new version with the schema already stored in the DSKB and detects the following update:

su_1 = *removeElement*(*Student*,{*phone*}), which specifies that the *phone* element was dropped from the type *Student*.

Subsequently, the SSIA identifies the following action to be executed in the DSKB to reflect the schema update su_1:

a_1 = remove all GCAs which reference the "phone" element from the type *Student*.

The SSIA identifies that the **GCA_3** (*student/phone* ≡ *member/phone*) was affected by the schema update and removes it from the DSKB.

Step 2: Update Notification

After updating the DSKB, the SSIA sends the following update notification to the MGMA: *u_n* = (*removeElement*(*Student*, {*phone*}), {*student/phone* ≡ *member/phone*}), specifying the schema update (*su_1*) and the set of global correspondence assertions ({*GCA_3*}) affected by this update.

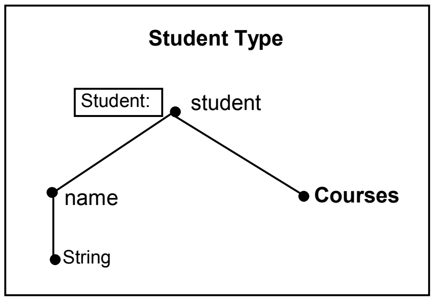

Figure 9 : New version of the student type of the Computer Science Department schema

Step 3: MKB Updating

When the MGMA receives the update notification *u_n* it determines the appropriate actions required to update the Mediator Knowledge Base (MKB). The updating of the MKB includes the updating of the mediator correspondence assertions and the updating of the mediator schema. In this case, the MGMA detects that the mediator correspondence assertion **MCA_3** (*$student_M/phone_M$* ≡ *student/phone*) was affected by the schema update *su_1* and must be dropped from the MKB. In accordance with the update notification *u_n* and the correspondence assertion **MCA_3** the MGMA detects that the mediator schema *StudentMember* was not affected by the update *su_1* and therefore does not need to be modified.

Step 4: Mediation queries Synchronization

Using the mediator correspondence assertion **MCA_3** and the global correspondence assertion **GCA_3** the MGMA executes the synchronization of the mediation query *$StudentMember_V$*. Initially, the MGMA verifies if the mediation query was affected and if necessary it performs the mediation query rewriting. In our example, line 7 of the *$StudentMember_V$* generated from the **MCA_3** must be redefined.

To rewrite the mediation query, the MGMA finds valid replacements for affected components of the existing mediation query. When a type is deleted, the MGMA attempts to find an appropriate substitute for the removed type. The MGMA searches in the set of global correspondence assertions specified in the update notification *u_n* for an appropriate substitute for the removed type. In our example, the **GCA_3** specifies that there is an appropriate substitute for the type phone. The **GCA_3** specifies that the set of *phone* elements from the type *Student* and the set of *phone* elements from the type *Member* are semantically equivalent. Therefore, line 7 of the *$StudentMember_V$* can be redefined as follows: <$phone_M$> \$m/phone </$phone_M$>, where the variable \$m is bound to a member *element*.

4 Related Work

Several data integration systems are described in the literature, including: TSIMMIS [8], SIMS [2], ARIADNE [1], MOMIS [4] and MIX [3]. Besides supplying data integration mechanisms some of these systems, such as the TSIMMIS, also offer tools to facilitate the data integration process. In TSIMMIS a common model, called OEM, and a specific query language, called LOREL, are used for data integration.

Other systems, such as the SIMS, were considered for integration of data stored in different databases, and later were customized for the web context. The adaptation of the ideas of the SIMS to the web gave origin to ARIADNE, a system for data extraction and integration from semi-structured web data sources.

MOMIS (Mediator envirOnment for Multiple Information Sources) is a framework for extraction and integration of structured and semi-structured data. An object-oriented language called ODL-I3, derived from the standard ODMG, is introduced for information extraction. Information integration is executed in a semi-automatic form, by exploiting the knowledge in a Common Thesaurus (defined by the framework) and ODL-I3 descriptions of source schemas with a combination of clustering techniques and Description Logics.

MIX is a wrapper-mediator system which employs XML as a means for information modeling and interchange between heterogeneous data sources. Mediator views are expressed in XMAS (XML Matching and Structuring Language), a declarative XML query language. To facilitate query formulation and for optimization purposes, MIX employs XML DTDs as a structural description of the exchanged data.

As mentioned earlier, one limitation of the majority of the data integration systems is related to the capability of evolving according to dynamic information systems. The work presented in [17] is one of a few to study the view adaptation problem in dynamic information integration systems proposing the Evolvable View Environment (EVE) framework as a generic approach to solve issues related to view evolution under schema changes for both view definition adaptation and view extent maintenance after synchronization. In contrast to our approach, EVE uses materialized views for data integration.

5 Conclusions and Future Work

In this work, we present an agent-based mediator system for data integration on the web. One important novelty of our system is that besides integrating data it uses agents to deal with the problems concerning generation and maintenance of mediation queries. We describe a set of tasks that must be performed by the agents in order to support mediator generation and maintenance.

We demonstrate how correspondence assertions specifying the semantics of XML-based mediators can be used to automate the maintenance of the mediation queries. We show through an example how to use the global correspondence assertions and the mediator correspondence assertions to help in rewriting a mediation query to reflect schema-level changes of data sources. One advantage of our approach is that we need to rewrite only the portions of the mediation queries affected by the schema update instead of having to generate the whole definition again.

As future work we intend to study the problem of having more than one possible mediation query definition from the same set of correspondence assertions. This problem is relevant for both the generation and for the maintenance of the mediated view. Moreover, as presented in [16] we also intend to analyze how to keep a history of the exported schema updates in order to minimize the impact of new updates in the integration system. Another important point that needs to be evaluated is the quality of the exported schemas which depends on the level of data source cooperation. The quality of the exported schemas is an important factor to determine the level of quality of the answers returned from the integration system.

References

[1] J. Ambite, N. Ashish, G. Barish, A. C. Knoblock, S. Minton, P. Modi, I. Muslea, A. Philpot and S. Tejada, Ariadne: a system for constructing mediators for internet sources. In *Proceedings of ACM SIGMOD Conf. on Management of Data*, pages 561-563, 1998.

[2] V. Arens, C. Y. Chee, C-N. Hsu and C. A. Knoblock, Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems*, 2(2):127-158, 1993.

[3] C. Baru, A. Gupta, B. Ludascher, R. Marciano, Y. Papakostatinou, P. Velikhov and V. Chu, Xml-based information mediation with mix. In *Proceedings of ACM SIGMOD Conf. on Management of Data*, pages 597-599, 1999.

[4] S. Bergamaschi, S. Castano, S. De Capitani Di Vimercati, S. Montanari and M. Vincini, A semantic approach to information integration: the momis project. In *Proceedings of Sesto Convegno della Associazione Italiana per l'Intelligenza Artificiale*, 1998.

[5] T. Bray, J. Paoli and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. http://www.w3.org/TR/REC-xml, Feb. 1998.

[6] A. Brown, M. Fuchs, J. Robie, and P. Wadler. XML Schema: Formal Description., http://www.w3.org/TR/xmlschema-formal/, Mar. 2001.

[7] D. Chamberlin, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu. XQuery: A query language for xml. http://www.w3.org/TR/xquery/. Feb. 2001.

[8] S. Chawathe, H. Garcia Molina and J. Hammer, The tsimmis project: integration of heterogeneous information sources. In *Proceedings of 10th Meeting of the Information Processing Society of Japan* (IPSJ), pages 7-18, 1994.

[9] S. Cluet, C. Delobel, J. Siméon, K. Smaga, Your mediators need data conversion!, In *Proceedings. of ACM SIGMOD Conference on Management of Data*, pages 177-188, 1998.

[10] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Suciu, A Query Language for XML. In *Proceedings of Eighth International World Wide Web Conference*, 1999.

[11] A. Elmagarmid, M. Rusinkeiwicz, and A. Sheth. Management of Heterogeneous and Autonomous Database Systems. 1^{a} Ed. Morgan Kaufmann Publishers, 1999.

[12] M. Fernandez, W. Tan, D. Suciu, SilkRoute: trading between relations and xml. In *Proceedings of 9th International WWW Conference*, pages 723-745, 2000.

[13] Y. Halevy, Theory of answering queries using views, SIGMOD Record, vol. 29, no.4, pp.40-47, 2000.

[14] T. Kirk, A.Y. Levy, Y.Sagiv, and D. Srivastava, The Information Manifold, in Proc. of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Environments, pp. 85-91, 1995.

[15] A. Y. Levy, Logic-based techniques in data integration, in J. Minker, editor Logic Based Artificial Intelligence. Kluwer Publishers, 2000.

[16] A. Koeller and E. A. Rundensteiner. A history-driven approach at evolving views under meta data changes. Technical Report WPI-CS-TR-00-01, Worcester Polytechnic Institute - Dept. of Computer Science, 2000.

[17] A. Nica and E. A. Rundensteiner, View maintenance after view synchronization. In *Proceedings of International Database Engineering and Application Symposium*, pages 215-213, 1999.

[18] A. Tomasic, L. Raschid, and P. Valduriez, Scaling access to distributed heterogeneous data sources with Disco, IEEE Transactions on Knowledge and Data Engineering, 1998.

[19] J. Shanmugasundaram, E. J. Shekita, R. Barr, M. J. Carey, B. G. Lindsay, H. Pirahesh, B. Reinwald, Efficiently Publishing Relational Data as XML Documents. In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB)*, pages. 65-76, 2000.

[20] V. M. P. Vidal, B. F. Lóscio and A. C. Salgado, Using correspondence assertions for specifying the semantics of XML-based mediators. In *Proceedings of WIIW 2001 - International Workshop on Information Integration on the Web - Technologies and Applications*, pages 3-11, 2001.

[21] J. D. Ullman, Information integration using logical views, in Proc. of ICDT'97, vol.1186 of LNCS, pp.19-40, Springer-Verlag, 1997.

[22] G. Wiederhold, Mediators in the architecture of future information systems. *IEEE Computer*, 25(3): 38-49, 1992.