MIROWeb: Integrating Multiple Data Sources Through Semistructured Data Types

Luc Bouganim, Tatiana Chan-Sine-Ying, Tuyet-Tram Dang-Ngoc,
Jean-Luc Darroux, Georges Gardarin, Fei Sha

MIROWeb Research Group, PRiSM Laboratory
University of Versailles Saint-Quentin
78035 Versailles Cedex, France
<Firstname>.<name>@prism.uvsq.fr

Abstract

The MIROWeb Esprit project has developed a unique technology to integrate multiple data sources through an object-relational model with semistructured data types. It addresses the problem of integrating irregular Web sources and regular relational databases through a mediated architecture based on a hybrid model, supporting relational, object and semistructured features. The project data exchange format is XML, the new standard of the Web, and the pivot language is XMLQL, a query language based on XML templates from AT&T. The demonstration will show the data warehousing approach for mediation, based on Oracle 8 and a semistructured cartridge developed in the project for supporting XML and XMLQL queries.

1. Introduction

Most distributed heterogeneous DBMSs adopt the so-called mediation architecture to federate multiple data sources. They integrate data from multiple heterogeneous sources and provide users with uniform integrated views of data. While the first generation of mediators was based on the relational model, the second was founded on some variations of the object model. It was illustrated by projects as IRO-DB [1], DISCO [2], Garlic [3] and Information Manifold [4]. A new generation, under development, uses a semistructured data model for dealing with the heterogeneity of the data sources. TSIMMIS [5] and YAT [6] are good representatives of this new trend.

Capitalizing on the previous IRO-DB experience, MIROWeb [7] uses a mediation architecture based on the object-relational model enhanced with a semistructured data type. With this approach, sources are modeled as tables with possibly semistructured attributes. Instances of those attributes are modeled as labeled directed graphs. Atomic objects are stored in object-relational tables. They contain values from one of the primitive object-relational types such as integer, real, string or user defined data types. They can be referenced as graph leaves of semistructured instances. This powerful yet simple model gives MIROWeb the capability of supporting both structured and non-structured data. It also greatly simplifies the development of source wrappers.

The choice of the object-relational model extended with semistructured objects as pivot model requires the use of a powerful object-relational DBMS at the mediator layer to integrate the data. The MIROWeb mediator is based on Oracle 8 which acts as the integration platform. Oracle 8 has been extended with a semistructured data type through the development of a Java cartridge to support semistructured objects.

To query semistructured objects, two solutions at least were possible: either build specific methods inside SQL3 or use a new query language more general than SQL3. MIROWeb retains the second alternative and chooses XMLQL [8] as pivot query language. Thus, any query

Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999.

exchanged between clients and mediators are XMLQL queries. Answers are XML documents. At the source layer, tables and complex objects (*e.g.*, HTML files) are translated in tables with XML attributes. At the mediator layer, XML documents are stored in tables with semistructured attributes. The mediator receives XMLQL queries from clients, decomposes them in XMLQL queries addressed to wrappers or in SQL queries for Oracle 8. Results are loaded in and processed by Oracle 8 extended with the semistructured cartridge. They are then transformed in XML to be sent to clients.

The demonstration will show the MIROWeb project in action on a toy integrated database. The database is derived from multiple sources: relational data, XML data and HTML files. Using the wrappers and loaders, the data will be loaded in Oracle 8 and its semistructured component. Then, through the user interfaces, we will browse and query the data in a uniform way using XMLQL.

2. System Architecture

As most mediation systems, MIROWeb is composed of three layers: client, mediator, and source. The client provides a *Query Browser* and a *Java API*. The mediator is based on Oracle 8. It includes the *Message Manager* to handle client-server protocols, the *Query Decomposer* and the *Semistructured Cartridge*. The system architecture is shown in Figure 1.

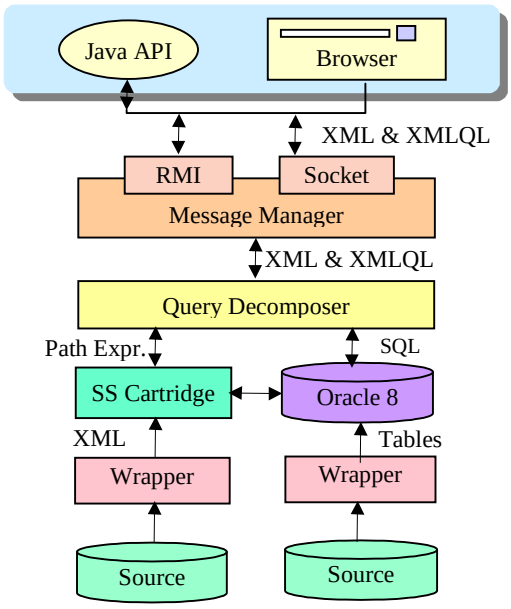

Figure 1: Overview of the MIROWeb System

The *Query Browser* is a graphical interface to browse through virtual XML documents from any starting label chosen by the user. To avoid errors, a list of accessible labels is fetched from the mediator dictionary, which contains a metadata structure similar to *dataguide* [9]. The user can choose a root, develop the hierarchy, formulate selection and join predicates, select projection nodes and submit the resulting XMLQL query.

The *Java API* is invoked by a client Java program through an extended JDBC interface to fetch data from the database. Basic extensions of JDBC are provided and allow to send an XMLQL query to the mediator and to navigate through the resulting XML document.

The *Message Manager* handles the client-server dialog. The clients communicate with the server via sockets or RMI. The message manager handles the two protocols. It also parses the query and delivers it to the query decomposer in an internal representation.

The *Query Decomposer* is responsible for decomposing queries into SQL queries for structured data and method calls to the Java cartridge to handle semistructured data at the mediator layer. The type of data (*e.g.*, semistructured or relational) and its status (*e.g.*, from which data source) is given by the dictionary through the dataguide. Queries are decomposed according to the referenced data type (semistructured or relational). When returned, results are recomposed as a unique XML document according to the query graph.

The *Semistructured Cartridge* implements semistructured objects within Oracle 8. From an external point of view, semistructured objects are seen as XML documents. Internally, they are modeled as labeled directed graph. However, our modeling is slightly different from data models such as OEM[10]. We distinguish two types of links: a link originated and targeted within the same object is called an *aggregation link*, and a link between different objects is called an *association link*. When assembling an object, only links of aggregation type are considered as relevant. This information is particularly important to model XML documents, which include M to N relationships (*i.e.*, association links). In addition, we introduce an optional order of edges among those going out from a given node. The order is simply memorized through an ordinal number. This is important to memorize and retrieve XML document items in the correct order.

In the current version of the system, all integrated data are first loaded within Oracle 8 in a data warehouse approach, then queried through the user interfaces. Storing the data is the responsibility of the mediator administrator through the wrappers and data loader. Later, it should be done on demand, at least partly.

3. Demonstration

We now introduce a typical scenario which will be demonstrated at the conference. Semistructured (XML)

real data are loaded within Oracle 8 using a specific database wrapper allowing to generate and rename metadata (*i.e.*, tags). Three data sources are first loaded in the mediator as tables with semistructured attributes. Then, the demonstration focuses on the browser and the mediator. It shows how XMLQL queries are formulated, then sent to the server, decomposed and processed either through the SQL engine or through the semistructured cartridge. In the following, we describe four typical steps of the demonstration.

The Control Panel

A user can access MIROWeb with his Internet Browser, going to the URL providing our services. Then, the MIROWeb Java applet is launched in the browser, and begins with a Control Panel (see Figure 2) where the user must be authenticated by giving *login* and *password*.
Once the user is authenticated, the dictionary window is shown.

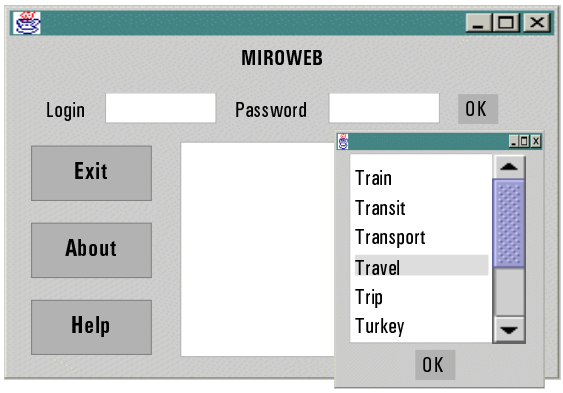

Figure 2: The Control Panel and the Dictionary

The dictionary

The Java applet first fetches all the labels which can be provided to the user. The labels are metadata from which the user wants to later start his query. These labels are generated from the *dataguide* built on the server as part of the dictionary, and sent to the client browser which parses and displays them as a list of labels. In our example (see Figure 2) the user has chosen the label *Travel*.

The Query Specification

MIROWeb is designed such that the user formulates queries graphically (see Figure 3). Specific buttons are provided to help the user.

- **Attribute**: clicking on this button switch to the attribute mode (default mode). In this mode, if the user clicks on a node (folder), then it will open/close it, connecting to the server if necessary to provide the next level of the opened folder. If the user clicks on a leaf, then an input form will appear on the screen, where the user can provide restrictions on the selected attribute. Filling it with a value means that the user wants this attribute correspond to the entered value. Strings, operators signs will be introduced. And filling it with a star (*) means that the user wants this value to exist, but does not care about the precise value.

- **New Root**: by clicking on this button and choosing a node, the user selects a new starting label for his query.

- **Projection**: in the projection mode, user can select/deselect attributes to be shown at result time.

- **Submit**: this button is used to submit the request to the server, and so the user will pass to the next step.

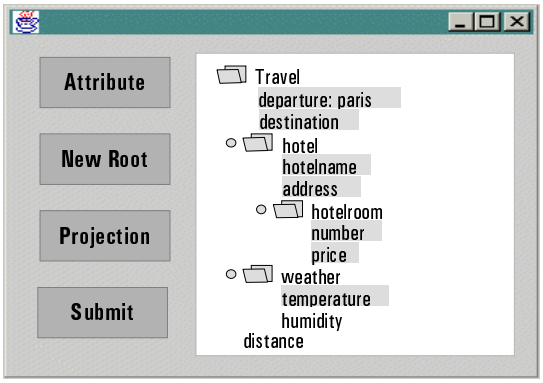

Figure 3: The Query Specification

In our example (see Figure 3), the user has formulated the following query:
select the departure, the destination of all the travels starting from "Paris" with the name and address of the hotel(s) for each corresponding travel, the number(s) and price(s) of each room of each hotel matching, and the weather temperature.

The Query Result

The query result (see Figure 4) is then generated by the server and sent to the client browser which displays it visually as a tree. As we can see on the screenshot, every matching result is represented as a node which can be opened if the user wants details on its content.

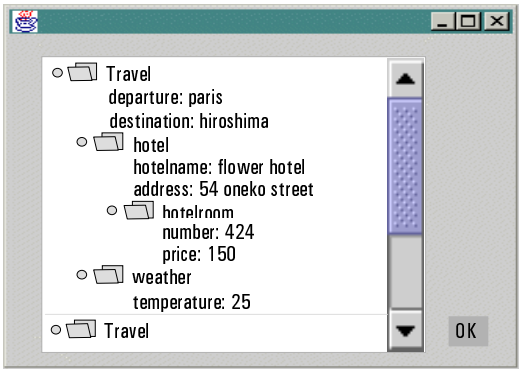

Figure 4: The Query Result

4. Conclusion

The development of MIROWeb shows a number of important points about the integration of data sources based on an hybrid object-relational semistructured model: (i) Object-relational systems can support quite efficiently semistructured objects based on a cautious implementation of graphs in tables using both clustering and indexing. (ii) From a user viewpoint, it is easy to browse through integrated data seen as XML virtual documents that can be queried on tags and contents. (iii) XML and XMLQL are appropriate and complementary exchange format and query language for heterogeneous irregular data. More than its power, the beauty of XMLQL is its homogeneity with XML, making queries as easy to transfer as documents. (iv) Decomposing XMLQL queries according to a repository giving the tag types and localization in SQL or XMLQL queries is possible. We have developed a special technology for that. (v) Developing intelligent wrappers capable of translating any source in a table with semistructured columns is quite easy. Furthermore, the wrappers can be intelligent enough to unify the tags based on a common directory. This last point requires further investigation.

5. Acknowledgments

We would like to thank Dana Florescu, Peter Fankhauser, Henri Laude and Patrick Valduriez for their help within the design of the MIROWeb system. Our research was sponsored by Esprit Project N° 25208.

6. References

[1] G. Gardarin, B. Finance, P. Fankhauser, W. Klas: "IRO-DB: A Distributed System Federating Object and Relational Databases", *In book "Object Oriented Multibase Systems: A Solution for Advanced Applications"*, Chap. 1, O. Bukhres and A. Elmagarmid Editors, 1994.

[2] A. Tomasic, L. Raschid and P. Valduriez: “Scaling Heterogeneous Databases and the Design of Disco”. *Int. Conf. on Distributed Computing Systems*, Hong Kong, 1996.

[3] M.J. Carey and all: "Towards Heterogeneous Multimedia Information Systems: The Garlic Approach", *IEEE Workshop on Research Issues in Data Engineering (RIDE-95)*, Taipei, 1995.

[4] A. Levy, A. Rajaraman and J. Ordille: “Querying Heterogeneous Information Sources Using Source Descriptions”. *Int. Conf. On VLDB*, Bombay, 1996.

[5] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman and J. Widom: ”The TSIMMIS Project: Integration of Heterogeneous Information Sources”. In *Proc. of 10th Anniversary Meeting of the Information Processing Society of Japan*, Tokyo,1994.

[6] S. Cluet, C. Delobel, J. Simeon and K. Smaga: “Your Mediators Need Data Conversion!”. *Int. Conf. on Management of Data ACM SIGMOD*, Seattle, 1998.

[7] P. Fankhauser, G. Gardarin and M. Lopez: “Experiences in Federated Databases: From IRO-DB to MIRO-Web”. *Int. Conf. on VLDB*, New York, 1998.

[8] A. Deutsh, M. Fernandez, D. Florescu, A. Levy and D. Suciu: ”XML-QL: A Query Language for XML”. *Submission to the WWW Consortium*, 1998.

[9] R. Goldman, J. Widom: “DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases”. *Int. Conf. on VLDB*, Athens, 1997.

[10] Y. Papakonstantinou, H. Garcia-Molina and J. Widom: “Object Exchange Across Heterogeneous Information Sources”. *Int. Conf. on Data Engineering*, Taipei, 1995