XLive: An XML Light Integration Virtual Engine

Tuyêt-Trâm Dang-Ngoc*, Clément Jamard**, Nicolas Travers**

* LICP Laboratory
University of Cergy-Pontoise
95302 Cergy-Pontoise CEDEX
dntt@dept-info.u-cergy.fr

** PRiSM Laboratory
University of Versailles
78035 Versailles CEDEX
(nicolas.travers, clement.jamard)@prism.uvsq.fr

1 Introduction

On the Internet, data are distributed on heterogeneous sources. To integrate them in a uniform view, lots of systems based on the famous mediator/wrapper architecture defined by [14] have been designed [3][12]. The data model now admitted for representing data is semi-structured data represented by the XML standard format. Thus now, as well in industry as in research, integration systems using XML-based standards have emerged [5][4].

XLive is such an integration system based on XML standards. It is the sequels of our experiences on mediation design in research project (MIROWEB [11], XML-KM) and in industry XMLMedia.

The XLive prototype is designed to be a light mediation system with high modularity and extension capabilities. It is a running research vehicle designed for assessing the integration system at every stage of the process starting from sources extraction to the user interface, including query parsing and modeling, optimization and evaluation, and also benchmarking.

2 System architecture

As most mediation systems, XLive is composed of three layers: Presentation, Integration and sources Connection.

Each of these layers is composed of several components that are all "exchangeable". Indeed, all components have a defined interface. It allows different implementations (for testing techniques or algorithms) on each component. Of course, all actions of components can be traced: viewing intermediate structures states, getting information on memory state and execution time.

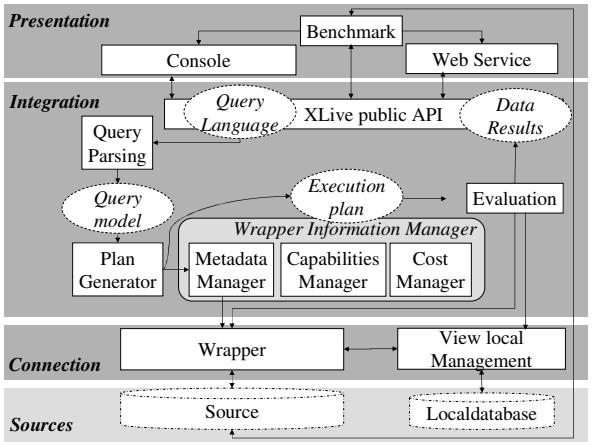

Figure 1: XLive Architecture

The integration system is shown in Figure 1. Components are shown as boxes and intermediate structures are shown as ovals. Structures and components can all be differently designed, but structures are bounded to the component. For example, for the XQuery query language we have two different Parsing Components, both parsing XQuery and for another query language; let's say XML-QL, we should use a dedicated XML-QL Parsing component. The XLive Integration System is composed of the following components:

- The *Wrapper* is a component for accessing a specific source for querying and retrieving result. As sources have specific access methods, the role of the wrapper is to translate the wrapper specificity to a common access method.
- The *Wrapper Information Manager* is for integrating information about wrapped sources. It provides to the mediator sources metadata, capabilities, and costs statistics on source data.
- The *View local Management* is a special source. It is used to define views that are implemented in two ways depending on data and source type.

- The *Query Parsing* parses a query language and translates it into a query model.
- The *Plan Generator* translates a query model to one or more execution plan. The optimal execution plan that would be evaluated is chosen by using cost information.
- The *Evaluator* evaluates the execution plan by querying relevant sources and merging results.

3 Modules

All components described above can be designed and implemented in different ways. Based on this modularity, XLive is used in different research projects (WebSI, SemWeb) by designing appropriate component to achieve project objectives. Thus, each component can be replaced by another implementation matching the same interface in order to test performance of different kind of algorithms and methods. In the following, we describe research algorithms and concepts that have been implemented and tested using the prototype. All implementation of XLive components (mediator and wrappers) is done in Java.

3.1 Evaluation Model

From a query language, the query parser designs a model for representing queries. Two models have been designed: RXQuery and TGV.

3.1.1 RXQuery and XQuery Rules

RXQuery is an intermediate structure for representing an XQuery. Each clause of the FLWOR expression of an XQuery is mapped to a specific structure (XFor, XWhere and XReturn). The following steps are then applied to create an execution plan from this model.

The first phase of decomposition transforms the query in canonical forms, i.e., without imbrications, using equivalence rules described in [13]. The RXQuery structure is transformed in several ones where imbrications are replaced by join constraints between flat RXQuery structures.

The second phase analyses the pending queries resulting from canonization to create atomic queries, i.e., requests processing only one collection that can be delegated to a wrapper. The join conditions and the final reconstruction allow the mediator to build the query plan.

3.1.2 Tree Graph View

The idea is to propose a representation of queries as graphs of trees, more precisely as tree pattern graphs interconnected by hyperlinks. The structure called *Tree Graph View* (TGV) [8] is an extension of the Generalized Tree Pattern graph proposed in [6] as a concise and practical representation of an XQuery. It is designed to be a more intuitive model of queries and to allow direct optimization before generating the physical execution plan.

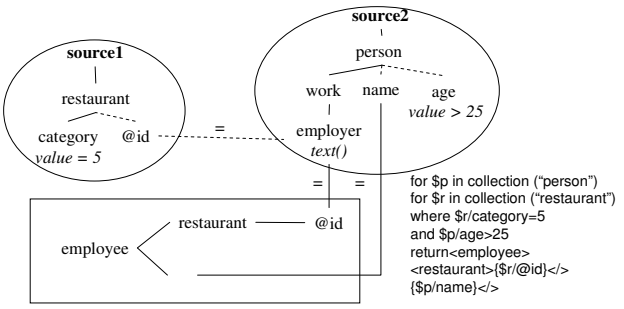

Figure 2: TGV representation for an XQuery

In Figure 2, the bottom right XQuery is mapped to the TGV graph on the left. Annotated Tree Patterns represents the request schema associated to sources and (hyper)links between them are used respectively for representing joins and reconstruction.

3.2 Execution plan

The mediator Plan Generator creates an execution plan from an evaluation model. The execution plan needs an algebra to represent operations to evaluate.

XAlgebra [7] is an extended relational algebra able to process XML trees in pipeline. The XAlgebra is designed to manipulate semi-structured data, thus it includes both relational operations to process tables of references on needed XPaths and navigation in XML trees. This representation is called XTuple; XOperators manipulate XRelations composed of XTuples. The XAlgebra is a physical algebra, which means that algebraic expressions are used to process XML flows and that algorithms are directly implementing them.

3.3 Wrappers

Wrappers translate sources from their native query language and result format to the common ones used by the mediator: XQuery and XML. Several wrappers have been implemented for XLive.

Relational Database Sources: Wrappers for relational databases have been implemented allowing the mapping from relational data to semi-structured data. They are available for Oracle and MySQL using JDBC.

XML Database Sources: Wrappers for native XML databases have also been implemented. They are available for XHive and Xyleme [2].
Web Service Sources: Two wrappers using web services provided respectively by amazon and google has been implemented. In the case of google, the wrapper sees the whole World Wide Web as a huge collection of documents, using the specific structure defined by google. In the case of Amazon, it is the whole amazon database that is seen as a collection.

3.4 View Local Management

The mediator integrates two view systems. The *Text View* integrates XQuery Text functionalities for non-capable sources. *Materialized Views* address the problem of interrogating web sources.

3.4.1 Text oriented indexed views

A module for creating indexed views on distributed sources is available for XQuery Text oriented queries. It uses an element identification scheme based on a structural guide of the view to determine the virtual position of indexed terms in the view. A global index maps a virtual document in the view to the local document in the source. On a query on the view, virtual documents identifiers answering the query are retrieved from the index, wrappers load values from sources using virtual-to-local document mapping and there is only to reconstruct results in the mediator. For better performance, we index references on distant data instead of materializing the view in the mediator.

This indexing system allows answering XQuery Text queries over distributed sources and bypasses the eventual lack of textual capabilities of sources. Moreover it also provides a uniform relevance ranking scheme, allowing choice between various formulas depending on research domain.

3.4.2 Materialized views for web sources

To optimize query evaluation on web sources, views are materialized in a local XML warehouse. The difficulty is to maintain incrementally the view as the sources change [1]. In the special case of web sources, maintenance is all the more difficult as web sources do not communicate any information about their update and about the data structure and ids.

In the context of web sources, very few information are provided by sources, and methods usually proposed cannot be applied in this context.

This component is for testing how to update materialized XML views on web sources in the context of mediation architecture.

In this approach, we introduce XTID in the XAlgebra which is a pair of number incrementally generated by the wrapper for each sources. By using XTuples comparison and XTID, we are able to report updates from the source to the view materialization by comparing diff fragments. More about it can be found in [1].

3.5 Wrapper Information Management

Information provided by wrappers are used in the mediator to validate queries, resolve sources and capacities.
Metadata: A generic Metadata Manager has been implemented based on path set. The path set describes each collection structure and associates each target sources. We now plan to implement another one based on XML-schema.
Capabilities: Wrappers provide capabilities rules to the mediator. The Plan Generator is able to understand these rules to optimize the execution plan by delegating as much as possible work to sources. Operations not supported by sources should be processed by the mediator and are included as operators in the execution plan. It leads to a valid execution plan, i.e., that can be evaluated at execution time. The mediator handles main operations (Join, Construct, Union, Sort …) and also text research operations.
Data information: Other wrapper information should be provided to the wrapper as plug-ins, like a generic cost model.

3.6 Benchmarking

A benchmark [10] created in the context of distributed semi-structured model on heterogeneous sources have been specified and applied to XLive.

It performs comparisons between queries evaluated on single sources and through the mediator. Mediator performance is shown in Figure 3 for a set of representative queries: selection and projection (q01-q02-q07), join (q09-q10), reconstruction (q11). We use two kinds of XML data in our scenarios, data oriented and structure oriented. XLive stores those data in different systems: native XML repositories (XHive, Xyleme), relational systems (Oracle, MySQL) and web sources (Google, Amazon). Generally the overhead due to the mediation process raises the execution time between 1.4 and 2 compared to single source evaluation.

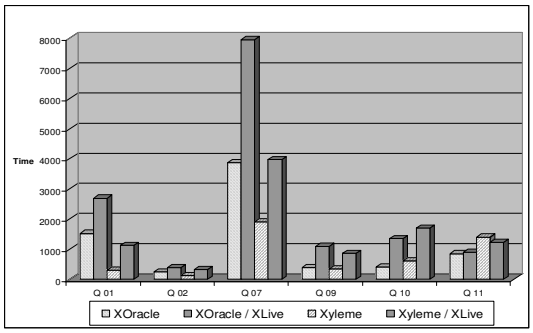

Figure 3 : Benchmarking (Oracle, Xyleme, XLive)

4 The Demonstration

The XLive prototype is now used in several projects and works entirely. It means that several heterogeneous distributed sources based on semi-structured data can be registered to the mediator and that an XQuery given to the mediator on these sources return the correct result into XML format in a reasonable delay.

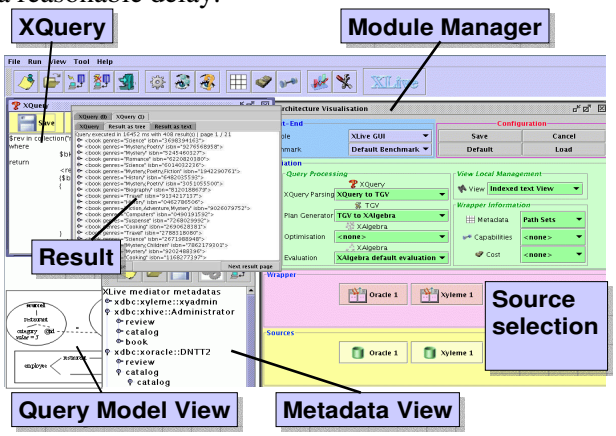

Figure 4 : XLive Graphical Interface

The demonstration will go through the execution process of an XQuery, from parsing to plan generation of a large set of queries on several distributed heterogeneous sources. Every step of the process is detailed allowing users to test and compare different query model, execution plans and evaluation. The graphical administration console displays intermediate structures (RXQuery, TGV, XAlgebra execution plan) for better process information.

The demonstration will show the ability to easily exchange components through our interface in Figure 4. We then compare performances of the mediator with different component combination (query and plan optimization, views management for XQuery Text).

5 References

[1] Abiteboul S. On Views and XML. PODS: 1999, pp 1-9, Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, USA.

[2] Abiteboul S., et al. The Xyleme Project. Computer Networks 39(3): 225-238 2002.

[3] Ahmed R., Albert J., Du W., Kent W. Litwin W., Shan M. An Overview of Pegasus. IEEE 1987.

[4] Baru C., et al. XML-Based Information Mediation with MIX. In Demo Session, ACM-SIGMOD'99, Philadelphia, PA, 1999.

[5] Chawathe S., et al. The TSIMMIS Project: Integration of Heterogeneous Information Sources, IPSJ Conf, Tokyo, Japan, 1994

[6] Chen Z., et al. From Tree Patterns to Generalized Tree Patterns: On efficient Evaluation of XQuery. Very Large Data Bases 2003, Germany Sept 2003.

[7] Dang-Ngoc T.T., Gardarin G. Federating Heterogeneous Data Sources. In proc of Intl Conf on Information and Knowledge Sharing (IKS), Scottsdale, USA, 2003

[8] Dang-Ngoc T.T., Gardarin G., Travers N. Tree Graph View: On Efficient Evaluation of XQuery in an XML Mediator In proc of 20th conf. Bases de Données Avancées (BDA), Montpellier, France, 2004.

[9] Dang-Ngoc T.T., Kou H., Gardarin G. Mediating the Web through XML concrete Views. In proc of Int Conf on DataBase and Applications (DBA), Innsbruck, 2004.

[10] Dragan F., Gardarin G. Benchmarking an XML Mediator, ICEIS 2005 Miami USA. To appear.

[11] Fankhauser P., Gardarin G., Lopez M., Muñoz J., Tomasic A.: Experiences in Federated Databases: From IRO-DB to MIRO-Web. VLDB 1998.

[12] Gardarin G., Finance B., Fankhauser P., Klas W. IRO-DB: a Distributed System Federating Object and relational Databases. OOMS 1994.

[13] Manolescu I., Florescu D., Kossmann D. Answering XML Queries over Heterogeneous Data Sources, 27th Intl Conf VLDB, Roma, Italy, 2001.

[14] Wiederhold G., Mediators in the Architecture of Future Information Systems, IEEE Computer, vol.25,N.3, 1992.