Integrating Web information with XML Concrete Views

Tuyêt-Trâm Dang-Ngoc Huaizhong Kou Georges Gardarin
PRiSM Laboratory, University of Versailles, France
+33(0)1 39 25 43 15
{Firstname.Lastname}@prism.uvsq.fr

ABSTRACT

To cope with the difficulties of Web information search, lots of technologies related to Web search engines have been proposed and have also seen very successful applications. Rather than yet another Web search engines with general purpose, this paper couples text mining and XML view caching techniques within Web mediation architecture and presents a prototype framework for topic-centric Web information search. Given a topic domain, domain-specific information is extracted from the Web documents belonging to the domain, then text-mining technologies are applied to discover the semantics contained in the Web information. Next we integrate the extracted information into a domain-specific common concept model defined using semantic Web languages. Finally an XML-based mediator allows the users to query the integrated Web information using XQuery. Once Web information is represented in the concept model with explicit semantic hierarchy understandable to the programs, user's queries against special fragments of Web documents can be carried out. One important part of our works aims at integrating XML view and cache techniques to manage Web information. Checksum technology is used to monitor the updates of Web page. One prototype is under construction centered on popular French sites of the finance domain.

Keywords

Web, Semantic Integration, XML, Mediator.

1. INTRODUCTION

Data integration and mediation are important technologies widely used in the information search applications that involve different information sources. As early as 1992, Wiederhold proposed a mediation architecture [13] for combining information from multiple data sources. Generally speaking, the mediation architecture consists of two types of components: wrappers and mediators. A wrapper is a software module built around an information source that logically converts the underlying data objects to a common information model [6]. A mediator is a software module that exploits encoded knowledge about certain sets or subsets of data to create integrated information for a higher layer of applications [13]. From then on, the mediation architecture is adopted by query-based information integration systems.

In the last years, faced to the increasing growth of information in electronic form accessible over the Web, the capabilities of efficient search and assimilation of Web information become more and more urgent. But compared to the case of searching information from multiple heterogeneous databases, searching from various Web sources is complex. In the first case, the major concern is the mismatch encountered in information representation and structure [13]. When integrating or combining various Web information, the major problems encountered include large and evolving number of Web sources, little meta-data available about Web sources, no clear Web source semantics accessible to applications, autonomy of Web sources, to name a few. To tackle these problems, XML-based technologies are increasingly used to describe and construct web sites. In the resource description area, major efforts are in progress under the Semantic Web activity at W3C (e.g., OWL). In the information search area, the classical mediation architecture has been adopted targeted at XQuery; XML views and caching techniques have been proposed for Web information integration [2][4][14].

In the framework of Web mediation architecture, we combined related existing technologies to provide topic-centric Web information search. Web wrapper technology is employed to extract domain-specific information from the Web interesting sites. Text mining techniques such as text categorization are applied to discover the semantics contained in the Web documents. The extracted Web information with their semantics are integrated into a domain-specific common concept model defined using semantic Web languages. An XML-based mediator called XLive is built on the top of the domain concept model, which allows the users to query Web information using the XQuery language. Once that Web information is represented in the concept model with explicit semantic hierarchy understandable to the programs, user's queries in XQuery against special fragments of Web documents can also be carried out thereof.

One important part of our works aims at developing efficient XML view techniques to manage Web information and exploiting cache techniques to optimize

information search performance over the Web. To optimize loading of source pages, lazy techniques are used to update concrete views. Timestamp, RDF description and checksum are used to monitor the updates of a Web page. XQuery view concretization accelerates information access. In the context of the project WebSI[1], we are constructing a prototype integrating some of these techniques. To evaluate the proposed system, we are federating some popular French sites of finance domain.

The rest of this paper is organized as follows. Section 2 describes the objectives and the architecture of our prototype. Section 3 introduces the mediator XLive to integrate and query distributed sources. Both Web wrapper technologies for extracting Web data and text mining technologies for semantics enrichment are discussed in Section 4. Section 5 introduces our principles of managing XML views. In Section 6, some related work is discussed and the main contributions of our work are outlined.

2. OBJECTIVE AND ARCHITECTURE

2.1 Objective

Traditional Web information search engines consider Web documents as simple character flows and neglect document semantics. User's query languages are mainly based on keywords. In this case, to respond a user's query the search engines search for query terms through entire documents and return the matching documents. Such keyword based full content search brings out some problems. For instance, it is not capable of taking advantage of the internal structures present in documents; it cannot return only document fragments containing relevant query terms; it cannot take into account the semantic context of document fragment.

Our objective is to develop a topic-centric semantic Web information search system rather than a general-purpose search system. Topic-centric search is quite natural, because the users have clear image of topics to be searched for. For this, given an application domain, a common ontology must be defined to express Web information belonging to the same domain. Some domain-specific semantics contained in Web documents have to be discovered so that Web documents be machine understandable. Machine understandable semantic structure can support information search at different abstract levels of documents.

2.2 Mediation Architecture

We use a mediation architecture to support web information integration and semantic search as shown in **Figure 1**. It follows the classical wrapper-mediator architecture as defined in [13]. The communication between wrappers and mediator follows a common interface, which is defined by an applicative Java or Web service interface named XML/DBC. With XML/DBC, requests are defined in XQuery and results are returned in XML format.

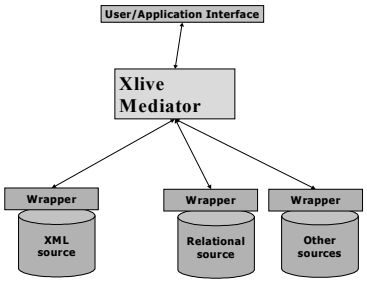

Figure 1. Architecture

Our architecture is composed of a mediator named XLive that deals with distributed data sources and wrappers that cope with the heterogeneity of the sources (DBMS, Web pages, etc.). The XLive mediator is a data integration middleware managing XML views of heterogeneous data sources. Using XLive one can integrate heterogeneous data sources without replicating their data while the sources remain autonomous. XLive is entirely based on W3C standard technology: XML, XQuery, XML-Schema, SAX, DOM and SOAP. All information exchanges rely on XML format. XML-Schema is used for metadata representation. Wrappers provide schemas to export information about local data structures. XQuery is employed for querying both the mediator and the wrappers. Connectivity of mediator and wrappers relies on the XML/DBC programming interface, an extension of JDBC to XML. More information about the XLive mediator can be found in [7].

To integrate a new source into the mediation architecture, a wrapper must be built to implement the XML/DBC programming interface, process some XQuery requests and return results in XML format.

DBMS are data oriented sources and metadata are provided to describe sources and mappings. DBMS wrappers translate data sources in XML and process a possibly reduced set of XQuery on the source data. In the case of Web source, the wrapper brings more intelligence. It aims at semantically integrating Web information in a common model accessible to programs.

3. SEMANTIC INTEGRATION

Semantic integration of Web information aims at transferring Web information at different Web sites from local source interfaces to a high level domain-specific common information model and making document semantic machine understandable. It is tri-folds, including the followings:

- Web data extraction: all domain-interesting information elements are extracted from related Web documents situated at various sites. Web wrapper technologies are employed to cope with such task. A Web wrapper is a

[1]Supported by the European Union 6th Framework Program

specialized program capable of automatically extracting data from Web pages and convert the results into semantically structured documents. Here the extraction rules are essential to the Web wrapper, which specify the location of certain interesting information elements in Web documents and also contain necessary instructions about how to map Web information fragments to concepts in domain-specific common data model. A wrapper can be generated using the smart toolkit SGWRAP [10].

- Text mining: text mining techniques are applied to analyze text content of Web information extracted at the precedent step and discover Web document semantics, which include topics discussed in Web pages, keywords, abstract, etc. For this, we have implemented text categorization machine learning algorithms, keyword extraction algorithm and statistics-based text summarization algorithms. Text categorization can assign to a document one or more than one predefined categories and can also find main topics contained in text document, then topics found may furthermore support the organization of Web information into domain-specific hierarchy of concepts, such as categorical organization of finance new stories at the Yahoo!Finance site. Text summarization provides an efficient way that allows user to get some insight into the content of Web documents. Automatic text-span extraction methods are often practiced to summarize document. The basic idea of text-span extraction is to extract some significant sentences from original document to build a summary. We adopted such extraction methods introduced in [11]. Semantics discovered by text mining are integrated within the common data model of application domain.
- Ontology representation: extracted Web information and their semantics are represented in a domain-specific ontology. Despite its population and power, XML has some limitations with regard to semantic representation. RDF is used to describe domain ontology. The single ontology approach [12] is taken to maintain one central domain ontology.

One software component called SEWISE has been developed to semantically integrate Web. See [10] for details on SEWISE.

4. WRAPPING THE WEB USING DOMAIN VIEWS

4.1 Motivation

Views techniques are mostly used to integrate distributed heterogeneous data [5]. Interesting works on views have already been done in the XML domain [1][14]. For the materialized views, incremental view maintenance strategies have been proposed [3][8].

Within the classical mediation architecture, one wrapper is built around each data source. In the Web context, since Web information are distributed at thousands of sites and pages, it would be too costly to instantiate one wrapper for each related Web source. One to one relation between wrappers and traditional data sources is not suitable for web information integration. Notice that existing Web pages can be classified into different application domains. Thus, to overcome combinatorial problems we can construct one or several views for each application domain. We build the domain views into one wrapper that let us transparently retrieve Web information from different Web sites of related domain. These views are created according to Web site contents and view definition. Views are used to optimize Web information access and process.

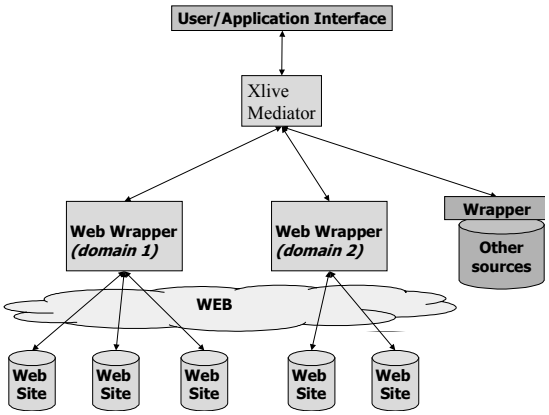

Figure 2. Web wrappers using domain view in mediation architecture

In addition, as the Web pages are very volatile and dynamic and loading them is costly, we need one solution to cope with complexity produced by volatile Web pages. To do so we create one wrapper to federate Web pages in the same application domain as shown in **Figure 2**.

According to update frequency of Web page contents, we can classify Web pages into three types:

1. **Static**: The pages are rarely changing. For example a personal Web page or the description of an enterprise.
2. **Semi-dynamic:** The pages are periodically updated and they can be preserved in a cache.
3. **Dynamic**: The pages are changed at each access time; it should be reloaded each time.

Downloading Web pages and processing them are expensive. There are four significant cost factors as follows:

1. **The time of downloading the Web page [9]**: Downloading time includes the processing time that the application server takes to deliver (static page) or generate (dynamic page, e.g., PHP or CGI) an HTML page plus the net communication time.
2. **The translating time**: the downloaded document can be of any type (HTML, XML, PDF, etc.). This step

consists of translating the document into XML using an appropriate translator following the type (such as TIDY for HTML).

3. **The processing time**: The processing is made by the SEWISE component It extracts semantics contained in Web documents and stores them in a final XML document.
4. **View computing time**: It is the execution time of an XQuery request against XML documents to answer a specified view.

According to the characteristics mentioned above, to optimize Web information search, we choose to materialize XML XQuery views and cache certain searched information. In general, we avoid to store all the pages in a cache (because of the dynamic pages) or to load from the Web all the relevant one on each request (because of downloading and processing cost).

4.2 Architecture

We propose the architecture given in **Figure 3** capable of managing views against static or dynamic Web pages and optimizing downloading and processing. We base our view manager on XML cache techniques to materialize the views on categorized Web pages. Our goal is to make this XML cache flexible enough to combine cache availability with Web connections to minimize the costs. This wrapper can be used with a mediator or in stand-alone. Its components are the followings:

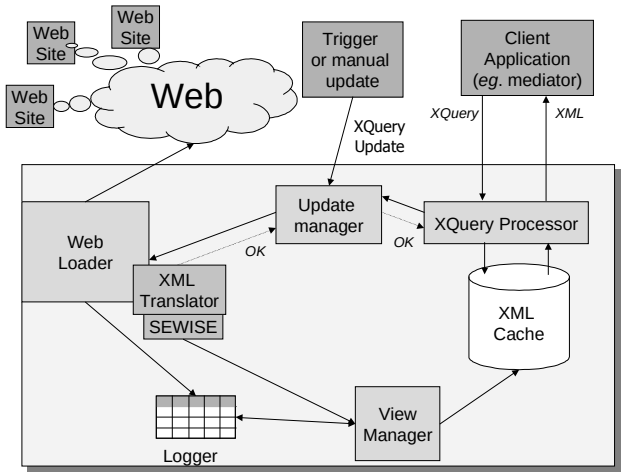

Figure 3. XQuery view architecture

4.2.1 XQuery Processor:

When receiving an XQuery request from a user or a client application (for example the mediator), the XQuery processor identifies the elements of collection needed for its evaluation. It asks the update manager for these elements. After the update manager sends back a confirmation, the XQuery Processor will carry out its request against the XML Cache.

4.2.2 Update Manager:

The update manager starts when it receives an update request from trigger or from the administrator (active case) but also maybe from a request of the XQuery Processor ("lazy" case). In both cases, it requests the Web loader to update documents and waits a confirmation from the later to answer the XQuery Processor.

4.2.3 Web Loader and Logger:

The Web Loader aims at loading Web pages on demand in the cases that the Web page has not yet been loaded and that the pages must be updated. Generally speaking, Web pages are loaded by the Web Loader (stage 1), then they are converted into XML by a translator associated with the file type (stage 2) and are passed to SEWISE that semantically analyze Web pages. For static pages (case 1), SEWISE processes the Web pages (stage 3) and passes them to the view manager that stores them (stage 4) in the XML Cache once time for all. For dynamic pages (case 3), the pages must go through the expensive process (stages 1, 2, 3 and 4) at each request. The most frequent cases is for semi-dynamic pages (case 2). In this case a lot of pages do not change since the last visit, so we need not go through all three processes (stage 1, 2, 3) that would be too costly.

We use a component named logger to store information about web pages. This logger keep information about the last time the page has been loaded (timestamp), description of the state of the page (RDF date or checksum) and a list of corresponding identifiers in the XML Cache.

If an RDF description is associated to Web pages, what the Web Loader must do is to load the RDF description and get the date information. Thus to know if Web pages have been changed since their last loading, we can simply consult the logger that preserves the timestamp of last changes.

Unfortunately in practice, RDF descriptions are rarely used to describe Web pages. In the most of cases it will be necessary to load whole Web pages and then compare them with their last loadings in order to decide if the pages are changed or not. One probably use a Web Cache (proxy) to store pages lately accessed. But this is very heavy (volume of accessed pages can be significant). Since we only need to know whether the pages are changed, so to do this we can only use the checksums of Web pages, which are kept in the Logger. For that, a checksum (as CRC32 or SHA) is calculated from the downloaded page. If the checksum is different from the one of the last time the page was downloaded, then it means that the page has changed.

4.2.4 XML Translator

It transforms received document into XML using a translator component tool specific to the file type of downloaded documents.

4.2.5 SEWISE Wrapper:

It analyzes HTML pages, and semantically represents them in XML documents that are passed to the View Manager.

4.2.6 View Manager:

The administrator initializes view using view definition requests. It receives XML documents from the SEWISE and executes the view definition to materialize them in the XML cache. At the first delivery of XML documents, it applies the XQuery view requests to them and stores the results in the XML cache . For the following delivery, it updates the XML Cache according to the received document and the already stored ones. The section 5 will further discuss XML views.

5. MANAGING XML CONCRETE VIEWS

We materialize the views in an XML cache. Materialization of the views makes it possible to provide a faster access to complex views, to optimize the processing time of a request and to improve data availability. We store materialized views in the XML cache.

XQuery has already a view mechanism enabled by user-defined functions. A function without any parameters corresponds to a non-parameterized view like SQL style, while a function with parameters corresponds to a parameterized view. With XQuery we can define XML Views using the following commands:

non-parameterized view:

```
define view FOO as
  query
```

parameterized view:

```
define function FOO ( ) as
  expression
```

The maintenance of materialized view is one of the main problems [3] [8]. In essence, one hypothesis for maintaining materialized view is that the view manager is informed about (by some trigger) the time when the sources are changed, the details of the change type (insert, delete, update) and the elements on which these changes are made (tuple relation, object ids). But in the case of views on a set of autonomous Web sources, the sources do not by themselves provide any information at update time.

We choose the lazy approach to update the materialized view only during evaluation of an XQuery request. So, it is not necessary for the source to inform the Web wrapper of eventual changes. In the same way, the Web Loader can use RDF date or checksum to decide if the Web source page is modified and then send a message to the view manager. It is the View Manager that is responsible of updating the materialized views in the XML cache.

The problem for the View Manager is to maintain the materialized XML View. One possible solution is that one reevaluates the view at each update, but that would be too expensive. We thus implement an algorithm to solve the update problem of view on distributed autonomous sources for simple views (that is, views without joins and aggregates).

Every node stored in the XML cache is identified by a pair (URL, node_id), where URL represents the Web page containing the node and node_id indicates the position of the node in the Web page. Without loss of generality, we can think of a web site as a collection of documents representing web pages. When we load a page, we can see if the page has nodes that have been changed, deleted or added.. The algorithm is as follows :

```
(1)  Coll_old <- loadOldCollection (Coll_new.id)
(2)  if (Coll_old == null) then
(3)     register (Coll_new.id)
(4)     for (doc_i in Coll_new) do
(5)        store (doc_i) in XMLCache
(6)     done
(7)  else do
(8)     for (doc_i in Coll_new) do
(9)        if (doc_i do not exist in Coll_old) then
(10)          register (doc_i) in CollectionInformation (Coll_old.id)
(11)          for (node_j in Coll_new) do
(12)             add (node_j) in XMLCache
(13)          done
(14)       else do
(15)          for (node_j in Coll_old) do
(16)             delete (node_j) in XMLCache
(17)          done
(18)          for (node_j) in Coll_new do
(19)             add (node_j) in XMLCache
(20)          done
(21)       fi
(22)    done
(23)    for (doc_i in Coll_old) do
(24)       if (doc_i do not exist in Coll_new) then
(25)          delete (doc)_i in CollectionInformation (Coll_old.id)
(26)          for (node_j in Coll_old) do
(27)             delete (node_j) in XMLCache
(28)          done
(29)          unregister (doc_j)
(30)    done
(31) fi
```

For each collection that has been changed, we load information about the old one (L1). If it does not exist (L2), the whole collection of documents is stored and referenced (L3, L4, L5). Otherwise, we must compare (L8, L23) documents of the new collection with the ones already stored in the XML cache by referencing their ID's. There are three cases: the document may be a new document (L9), the document already exists in the database and is probably modified (L14), and a document of the XML cache has been deleted (L23). In the first case, the document is registered (L10) and stored (L11); in the second case, the nodes of the old document are replaced by the nodes of the new one (L15-20) ; and in the last case, the document is unregistered (L29) and the associated nodes are deleted (L26).

In summary, the proposed strategies can be applied to maintain the materialized views in various cases. The view manager can only manage simple views but needs no complex operations like nested views, aggregates, products, etc. These kind of operations can be carried out by higher level user programs. For example, in the mediator architecture, instead of the wrapper, the mediator can do these operations.

Due to distributed sources and the tree structure of XML documents, defining complex view with join or aggregate would be difficult [3] and moreover maintaining dependencies between document collections from different autonomous sites would be very costly. At present, we only define simple views in our implementation.

6. RELATED WORK AND CONCLUSION

In [13] the Weaves Management System has been described, which relies on the declarative specification of Web Sites. This system describes a customizable cache system that implements the optimal data materialization strategy according to the Web site's specifics. They materialize relational, XML and HTML data to generate Web pages. Investigations have also been done about the Web caching [4] of HTML, and more recently XML. Web wrappers for XML mediation and semantic Web integration have been developed in the framework of TSIMMIS [6]. The Xyleme XML warehouse [1] provides an interesting approach to cache the Web. Xyleme discovers XML pages on the web that are of interest for customers. For the Web crawler of Xyleme crawls the Web (HTML+XML) and maintain up to date copies in the warehouse. Xyleme provides efficient algorithms to determine what pages to refresh. Our approach is more a mediating approach. It is original in the sense that it considers mediating information from semantic Web by using concrete XQuery views for optimization. It also integrates multiple techniques such as mediation, text mining, Web wrappers, and concrete views to support topic centered Web search.

More generally, in this paper we have shown how to optimize a wrapper for semantic Web integration in a mediation architecture. The "one wrapper for one source" schema used in classical mediation architecture is not exactly suitable for Web sources due to the problem of performance. Techniques to define XML views and maintain them have also been presented. We design Web wrappers as "small mediators" to build a unified XML view of different sources. We have also described how to integrate such a wrapper in a full-XML mediation architecture.

Currently, we are coupling the existing components in the context of the WebSI project. We shall report on our work progresses in future papers.

7. REFERENCES

[1] Abiteboul S., Cluet S., Ferran G., Rousset M.C. *The Xyleme project.* Computer Networks 39(3), 2002

[2] Abiteboul S. *On Views and XML.* PODS: 1999

[3] Abiteboul S., McHugh J., Rys M., Vassalos V., Wiener J.L. *Incremental Maintenance for Materialized Views over Semistructured Data.* VLDB pp.38-49, 1998.

[4] Barish G. and Obraczka K. *World Wide Web Caching: Trends and Techniques.* IEEE Communications Magazine Internet Technology Series, May 2000.

[5] Chaudhuri S., Krishnamurthy R., Potamianos S., Shim K. *Optimizing queries with materialized views.* Proc. of IEEE Conf. on Data Engineering, 1995.

[6] Chawathe S., Garcia-Molina H., Hammer J., Ireland K., Papakonstantinou Y., Ullman J., and Widom J. *The TSIMMIS Project: Integration of Heterogeneous Information Sources.* In Proc. of IPSJ Conf., pp.7-18, 1994

[7] Dang-Ngoc T.-T. and Gardarin G. Federating heterogeneous data sources with XML. In Proc. of IASTED IKS Conf., 2003

[8] El-Sayed M., Wang L., Ding L., Rundensteiner E.A. *An algebraic approach for incremental maintenance of materialized XQuery views.* WIDM 2002: 88-91

[9] Feldmann A., Caceres R., Douglis F., Glass G., and Rabinovich M. *Performance of Web proxy caching in heterogeneous bandwidth environments.* In Proc. of the IEEE Infocom '99 Conf., New York, 1999.

[10] Gardarin G, Kou H, Zeitouni K., Meng X. and Wang H., *SEWISE: An Ontology-based Web Information Search Engine*, 8th NLDB, Germany, June, 2003

[11] Goldstein J., Kantrowitz M., Mittal V.O., and Carbonell J. *Summarizing Text Documents: Sentence Selection and Evaluation Metrics.* 22th ACM SIGIR, CA, pp.121-128, 1999

[12] Wache H., Vogele T., Visser U., Stuchenschmidt H., Schuster G., Neumann H. and Hubner S. *Ontology Based Integration Approaches*, IJCAI Workshop: Ontologies and Information Sharing, pp.108-11, 2001

[13] Wiederhold G., *Mediators in the Architecture of Future Information Systems*, IEEE Computer, vol.25,N.3, pp.38-49, 1992

[14] Yagoub K., Florescu D., Issarny V., and Valduriez P. *Caching Strategies for Data-Intensive Web Sites.* In VLDB Journal, pp. 188-199, 2000