An Efficient Data Extraction and Storage Utility For XML Documents

I. Budak Arpinar[1], John Miller[2], and Amit P. Sheth[3]

LSDIS Lab, Computer Science Department
University of Georgia, Hardman Hall, Athens, GA – 30602
[1] budak@ainge.cs.uga.edu, [2] jam@cs.uga.edu, [3] amit@cs.uga.edu

1 Introduction

In this paper, a mechanism to provide selective extraction of data objects from XML documents, the storage of these documents in an object-relational database, and retrieval and reconstruction of XML documents from extracted data objects is discussed. The motivation is provided by a need for a Workflow Process Repository in a Workflow Management System (WFMS) [6], namely METEOR WFMS, to store meta-data about workflow designs, organizations, informational resources and computational resources. Thus meta-data is composed of different XML documents representing different components of a workflow process.

The repository, involving the Data Extraction and Storage Utility (i.e., Extractor), has the following main capabilities:

- Filtering of XML objects that need to extracted,
- Generating relational schemas for on-the-fly storage of XML documents,
- Loading data from XML documents into relational tables,
- Re-creating original XML documents as needed,
- Querying, browsing, and versioning.

Our XML storage scheme is so practical and flexible. Practicality comes from the broader acceptance and use of Object-Relational Database Management Systems (ORDBMSs); flexibility is provided by selective extraction mechanism (i.e., filtering) employed by the Extractor, which is not available in similar approaches [3] using an ORDBMS. The comparison of our approach with XML databases (e.g., Lore [5]) in terms of efficient storage and querying XML documents [1,2] requires more research and performance testing. Although they support a native data model for XML documents, many XML databases do not provide high performance.

Recently, some database vendors have started to support XML. Axielle by Ardent Software, eXcelon by ODI, and XML Repository by Poet are some examples. In general, these products provide an import and export programming interface, which is compliant with Document Object Model (DOM) from the W3C.

2 Repository Architecture

The motivation behind building an XML based workflow repository to support multi-organizational workflow processes, as well as to support reusability, adaptability and survivability of both intra- and inter-organizational workflows. Multiple organizations on the Web can post their services into the repository as workflow steps, and these steps can be incorporated into other organizations' workflow processes using repository's querying and browsing capabilities. Even the organizations can use their own local repositories to reuse existing workflow components, eliminating design of new workflows from scratch. Finally, the survivability is supported by replacing failed workflow components with functionally equivalent components at run-time, thus changing workflow schemas on the fly. These new components are searched in the repository using the graphical query composer and placed into the new workflow schema by a "drag-and-drop" [4].

The workflow repository architecture is depicted in Figure 1. Dynamic XML (DXML) from ObjectSpace provides ability to access and manipulate XML objects like regular Java objects. The Document Type Definitions (DTDs) are processed to generate classes for each element defined in a DTD. XML objects that need to be extracted, as well as mapping between relational tables/attributes and these objects, are defined in a *spec* file. Relational schemas are created dynamically using this specification, and extractor uses generated classes and Java reflection to retrieve XML objects to be stored in the corresponding relations. The XML documents themselves are stored as Character Large Objects (CLOBs), and the extracted information is correlated with corresponding CLOBs using foreign keys. In this way, extracted information can be queried (or browsed), and once desired object is located, original XML documents (e.g., workflow schemas) can be reproduced from the database.

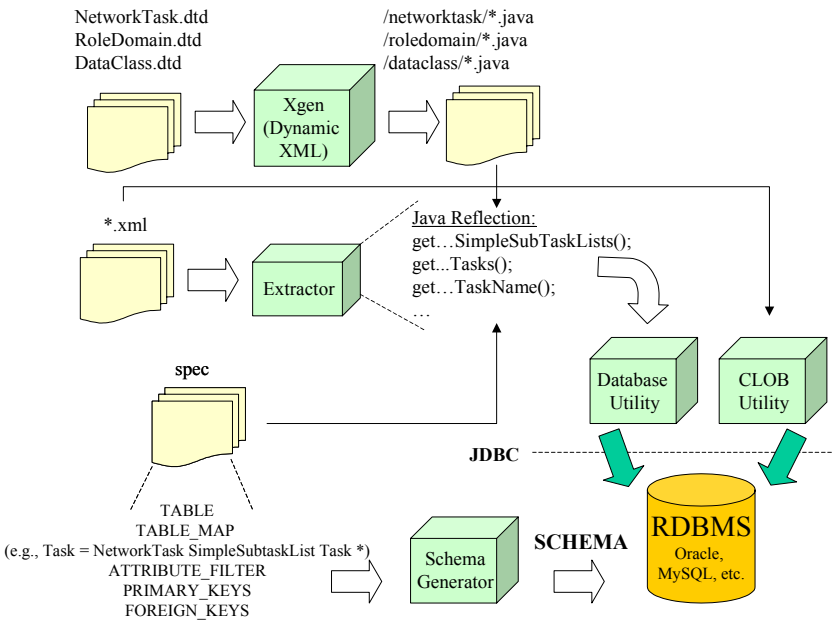

Figure 1. An Overview of Repository Architecture

3 Data Filtering and Extraction From XML Documents

An XML-relational mapping scheme is used to create a relational schema corresponding the "filtered" hierarchy of an XML document. Actually, both an XML document and a relational database can be viewed as trees. In Figure 2, a tree representation of the XML document below, namely *SampleFlow.xml* describing workflow steps (i.e., tasks) is depicted.

```
<?xml version="1.0"?>
<!DOCTYPE NetworkTask SYSTEM "NetworkTask.dtd">
<NetworkTask id="1">
  <Task id="2">
   <Name>SampleFlow</Name>
   <TaskType>Non-transactional WorkFlow</TaskType>
  </Task>
  <SimpleSubTaskList>
     <Task id="3">
        <Name>Start</Name>
        <TaskType>Human</TaskType>
     </Task>
     <Task id="4">
        <Name>Close</Name>
        <TaskType>Transactional</TaskType>
     </Task>
  </SimpleSubTaskList>
</NetworkTask>
```

Filtering is realized by (1) mapping only "selected" *elements* to *relations*, and (2) mapping only "selected" *elements* or *attributes* to *relational attributes*. For example, only `NetworkTask` and `Task` elements (Figure 2) are mapped to some relations. The property (2) above is realized by filtering only a selected subset of attributes associated with the leaf nodes of the paths starting with each filtered relation. For example, the extracted elements and attributes are depicted with dashed lines in the figure. Extractor recursively traverses XML tree structure from the Root downward using the interfaces and classes generated by DXML to extract specified attributes for each relation. Note that for brevity, a complete extraction algorithm is not discussed here.

The two employed filtering techniques ((1)&(2)) allow users to control creation of database schemas and provide an efficient storage mechanism for the selected components, because instead of a whole document, important parts of the document for a user are specified and extracted.

The resulting relational database is depicted in Figure 3. Filtered elements (i.e., `NetworkTask` and `Task`) are mapped to relations, and filtered elements/attributes (e.g., `Name` and `TaskType`) are mapped to attributes. A special relation, namely `XmlDoc`, is used to store XML documents themselves in `DocumentText` attribute as CLOBs. A `VersionNo` attribute is placed to support different versions of the same XML document.

As a final step, foreign and primary keys are computed for each relation. A foreign key is used to represent ancestor-successor (i.e., containment) relationship in the XML tree. In this way, attributes of ancestor nodes, or even attributes of sibling nodes can be accessed by using these foreign keys as join attributes. For each relation, the relation(s) referenced are indicated and foreign key(s) are produced to reference that relation(s)'s primary keys. A generated name for a foreign key is "<relation-name>_<primary-key>" of what it references. For example, for `Task` "`XmlDoc_DocumentId`" references the XML

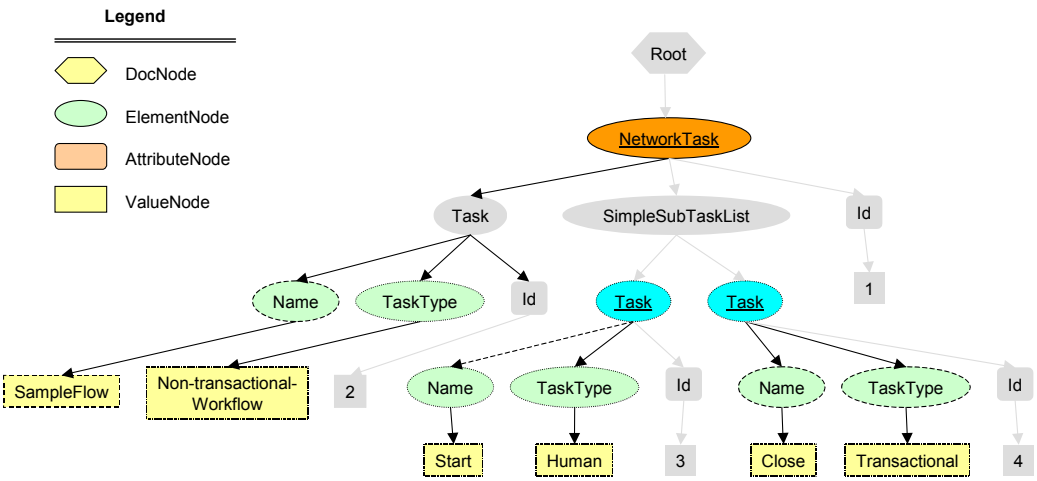

Figure 2. Filtering Elements and Attributes

Document. In general, a filtered XML element in the tree can reference one of the primary keys of a parent element, which is also filtered, or XML document to which it belongs. Finally, the primary keys are specified and they are underlined in Figure 3.

4 Conclusions

Recently, XML gains a great acceptance as a data interchange format on the Web. Thus, providing storage and querying capabilities for XML attains interests of many researchers. However, a broadly accepted solution is still missing. We believe that our approach provides for a flexible and practical solution until XML DBMSs are improved and standardized. Furthermore, the XML based workflow repository provides easy exchange of workflow process definitions between companies, and an integration tool to enable coordination of companies' business processes.

References

[1] "The XML Query Algebra", W3C Working Draft, Dec. 2000.

[2] S. Abiteboul, P. Buneman, and D. Suciu, "Data on the Web", Morgan Kaufmann, 2000.

[3] R. Bourret, C. Bornhovd, and A. Buchmann, "A Generic Load/Extract Utility for Data Transfer Between XML Documents and Relational Databases", WECWIS 2000, Milpitas, CA.

[4] M. H. Kang, J. N. Froscher, A. P. Sheth, K. Kochut, and J. A. Miller, "A Multilevel Secure Workflow Management System", CAiSE 1999: 271-285

[5] L. McHugh, S. Abitebul, R. Goldman, D. Quass, and J. Widom, "Lore: A Database Management System for Semistructured Data", ACM SIGMOD Record, 26(3), 1997.

[6] A. Sheth, W. Aalst, and B. Arpinar, "Processes Driving the Networked Economy: Process Portals, Process Vortexes, and Dynamically Trading Processes, IEEE Concurrency, July-September 1999.

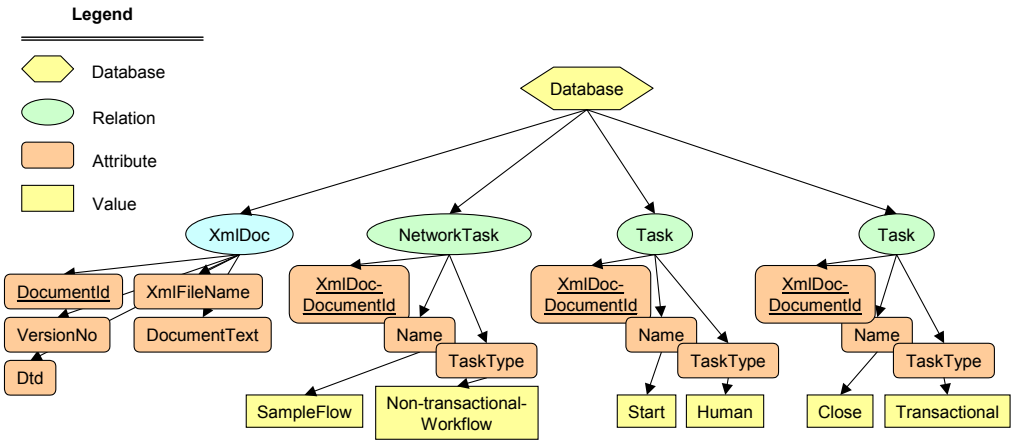

Figure 3. Dynamically Created Relational Database