Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp481-485)

XML based Framework for ETL Processes For Relational Databases

TASSAWAR IQBAL, NADEEM DAUDPOTA
Department of Computer Science
COMSATS Institute of Information Technology,
Abbottabad, NWFP
PAKISTAN

Abstract:- In Data Warehousing, Extraction-Transformation-Loading (ETL) are the key tasks that are responsible for the extraction of data from several sources, their cleansing, customization and insertion into data warehouse [10]. More specifically ETL tools are category of specialized tools with the task of dealing with data warehouse cleaning and loading problems. These task are very critical in every data warehouse environment, It is observed that ETL and data cleaning tools are estimated to cost at least one third of effort and expenses in the budget of the data warehouse [1,11], another evidence shows that ETL process costs 55% of the total cost of the data warehouse [1,12]. In this paper, we focus on the problem of the definition of ETL processes using xml in order to make this framework more generic and capable to deal with heterogeneous source systems. We described the framework that extract data from various heterogeneous source systems and carry it in xml files, later on data cleaning is performed using few predefined xml templates, predefined functions and ultimately data is loaded into data warehouse as per warehouse schema.

1. Introduction

Data warehousing systems integrate information from Transaction Processing Systems (TPS) into a central repository to enable analysis and mining of integrated information but it can be only achieved when we have all possible information from various TPS, if it is standardized and cleaned. Information must be with no missing values, no extra and varying symbols, no inconsistent codes and duplicates. Normally customized applications are used to perform this task that are designed keeping in mind the source system as well as destination systems structure and hierarchy. In this proposed framework extraction is performed using component that is customized in nature for each source database named extractor that extract the data from relational databases using SQL command and present it in the xml document. Second module named Cleansing Engine operate on these xml document and by using some predefined xml templates based on business rules it standardize and clean the data. Finally the standardized and cleaned data is mapped to the central repository of warehouse through Mapping Engine. Complete model is shown in figure 1.

Significance of this model lie in introduction of two generic components named Cleansing Engine and Mapping Engine that are designed in such a way that user can customize them through few simple options at interface level to define the business rules and to achieve the desire results.

The rest of the paper is organized as follow: Section 2.1 describes the Extractor Component in detail, Section 2.2 explains the Cleansing Engine of this model whereas Section 2.3 describes the Mapping Engine of this conceptual model and Section 3 brief overview of the GUI. At end section 4 consist of conclusion & section 5 is about references.

2. Framework

2.1 Extractor

This component of the model is an initiator that is responsible to extract data from various TPSs. This component has as many sub units as many TPSs exist in the data warehousing environment. Each sub unit is customized for particular TPS needs and architecture. Each sub unit extract the data from respective TPS and convert that data into the xml format that is standard in this model to make it generic.

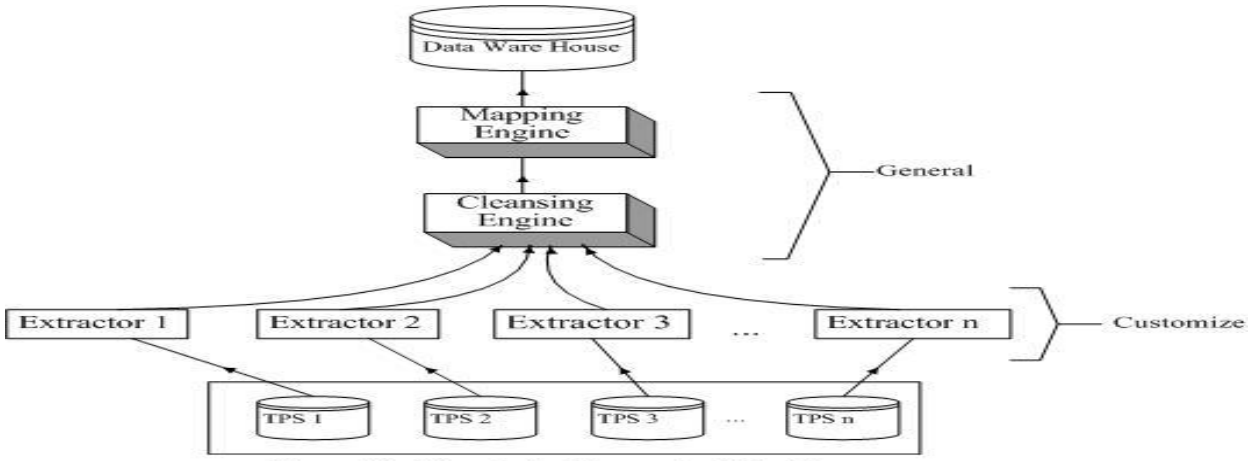


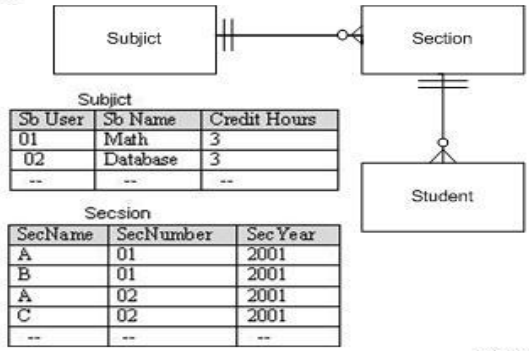
Figure 1 : Complete Conceptual Model

Let the one of the extractor deployed to extract data from a TPS where result record keeping is being managed as shown in the figure 2. This extractor sub unit application is customized in such a way that, it extracts the data using SQL commands and converts it into the xml document and makes this document available to the Cleansing Engine as input to perform the_cleansing operation at generic platform. Xml schema documents also provided with xml instant document or xml documents as in figure 5 & 6. These xml schema documents are actually those documents that were created by the customized Extractors those have complete information about a related xml document.

An xml document generated by this sub unit application is shown in figure 5. This document is organized in such a way that it has not only attribute name with their corresponding values but also has other necessary information such as type of attribute, length of attribute, either it is primary key or foreign key, attribute is not null constrained or not. This information is stored in tagged attributes of the xml documents and will be helpful in cleansing operations in later stages.

2.2 Cleansing Engine

This component of the framework is generic in nature. It means, it is flexible enough to clean data by simply adjusting some parameters through a user friendly interface that are part of this proposed framework. Cleansing Engine is provided with various xml templates with corresponding schemas shown in figure 6. Each template is being generated through user instruction specified through the interface, where each template is responsible to perform the cleansing against particular error.



Subjict

Sb User	Sb Name	Credit Hours
01	Math	3
02	Database	3
--	--	--

Secsion

SecName	SecNumber	SecYear
A	01	2001
B	01	2001
A	02	2001
C	02	2001
--	--	--

Student

RegNumber	SecName	SecNumber	StName	StCalss	StGrade
R1	A	01	Ali	BS	B
R2	A	02	Babar	BS	A
--	--	--	--	--	--

Figure 2 : Students result system

As shown in figure 3, the document generated from relational databases using extractor is provided to this component with specification that what kind of cleansing operations are required. Here XML templates can also be used for extra instruction for a particular document during cleansing. However various cleansing functions are provided in the function library of the cleansing engine. These functions are designed by keeping in mind all possible cleansing errors. Errors may be of types such as missing records, wrong codes, wrong information entered in the source system, duplicate records and unnecessary symbols [2].

Various methods are already devised to cope with these errors [3]. These methods are following.

- N-gram Sliding window for Spell errors
- Missing values determination
- Outlier for extra symbols detection.
- Inconsistent code detection
- Duplicates detection
- Name and Address cleansing

Details about each method are available in [3]. Lot of other methods and algorithms are already devised can also be used [4, 5].

Library contains the functions based on these pre-devised methods. Cleansing engine calls these functions and pass as parameter, the xml document, related schema and xml template if required. This cleansing process continues in pipelined fashion as shown in figure 4. This pipelined fashion helps to save time using parallelism.

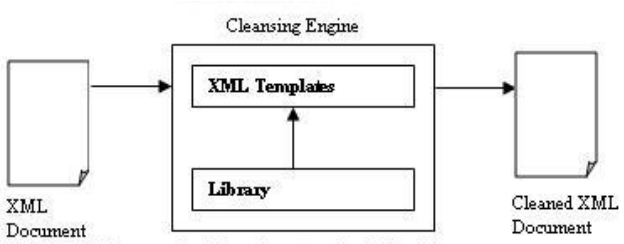


Figure 3 : Clensing Engine Module

2.3 Mapping Engine

Mapping the data from source to destination demands series of activities, but hard coded modules in various models make this process very rigid [9]. However in our proposed model, it is tried to overcome these issues.

All the cleaned xml documents are handed over to the last component of this framework called Mapping Engine. This component is also instructed by the user interface to map the data to destination. It map the xml documents to the destination with the help of xml schema, it collects all information (about relationships, attributes, their types and field length etc) of elements (named entities in relational databases) from schema and map them on data warehouse. But this is not an easy task, in order to achieve it accurately; mapping engine is instructed in detail, about the kind of model being used in data warehouse, about all possible calculations and aggregation function that are required on data that is available in xml document. Once all constrains are defined, using the SQL commands with the support of the X-Query commands (used for xml operations) documents are mapped to the data warehouse. Other query languages for xml documents can also be used such as UnQL [6, 7].

Spell-Error	✓					
Missing Record		✓				
Outlier Detection			✓			
Inconsistent Code Detection				✓		
Duplication Detection					✓	
Name & Address crossing						✓

Figure 4 : Pipeline Clensing Process

Let us assume that a star schema is being used in data warehouse that is final destination of the data, mapping engine map the xml document files into target schema by separating the master element for fact table and detail elements for the dimensional tables [8] using specified querying languages.

3. Support of GUI for Cleansing Engine & Mapping Engine

As mentioned earlier that significance of this model lies in a fact that it provides the cleansing and mapping operations using a non-customized platform through user-friendly graphical user interface. The xml documents that extracted from any relational database (it may be Oracle, SQL Server and Access etc) can be now operated on this non-customized platform. The model will treat each document without any distinction of its RDBMS because now it's in generic xml form. Model makes this process simplest by introducing a graphical user interface where user can demand the cleansing operation from the application through few simple instructions at GUI. Likewise the mapping can also be provided through the same GUI through some simple instruction at same interface.

4. Conclusion

In this paper we have focused on the problems of definition of ETL stages and provided a framework for their conceptual representation. We have proposed a novel conceptual model that is using xml support and generic in nature. Most specifically user can specify all the cleansing and mapping rules using GUI. Designer and programmer can establish a generic cleansing and mapping engine using any 4GL. It may be possible that performance is degraded but main feature of this proposed model is generic handling of critical issues of cleansing and mapping.

No doubt performance has been very critical issue in each application. But it is also a known fact that generality always affect the performance. That's why every non-customized product has lesser performance than special purpose product. However in case of this model, approach of pipelining mention in section 2.2 can be used to exploit parallelism that can compensate the performance to some extent, also indexing can be used to improve the performance while processing the xml documents for specified operations.

Application that would be developed using this model must be generic enough that covers almost all cleansing and mapping requirement, but this can never be done in single step. An iterative enhancement of application with increasing needs can make a perfect

non-customized commercial application to perform these tasks.

```
<?xml version="1.0" standalone="no"?>
<result>
<subject>
<sbnumber>01</sbnumber>
<sbname>Math</sbname>
<credithours>3</credithours>
<section>
<secnumber>A</secnumber>
<secyear>2001</secyear>
<student>
<regnumber>r1</regnumber>
<stname>Ali</stname>
<stclass>BS</stclass>
<stgrade>poor</stgrade>
</student>
</section>
</subject>
----
----
</result>
```

Figure 5 : XML Instant Document

```
<xsd:element name="result">
<xsd:complextype>
  <xsd:sequence>
    <xsd:element name="subject" type ="Subject" minOccur="0"
        maxOccur="unbounded"/>
    <xsd:element name="section" type ="Section" minOccur="0"
        maxOccur="unbounded"/>
    <xsd:element name="student" type ="Student" minOccur="0"
        maxOccur="unbounded"/>
  </xsd:sequence>
</xsd:complextype>

<xsd:key name="sbnumber">
   <xsd:slector xpath="subject">
   <xsd:field xpath="sbnumber">
</xsd:key>
<xsd:key name="secnumber">
   <xsd:slector xpath="section">
   <xsd:field xpath="secnumber">
</xsd:key>
<xsd:key name="stnumber">
   <xsd:slector xpath="student">
   <xsd:field xpath="stnumber">
</xsd:key>
<xsd:keyref name="secnumber" refer="secnumber">
   <xsd:slector xpath="section">
   <xsd:field xpath="secnumber">
</xsd:keyref>
<xsd:keyref name="stnumber" refer="stnumber">
   <xsd:slector xpath="student">
   <xsd:field xpath="stnumber">
</xsd:keyref>
<xsd:element name="subject">
   < xsd:sequence>
      <xsd:element name="sbnumber" type="xsd:unsignedInt"/>
      <xsd:element name="sbname" type="xsd:string"/>
      <xsd:element name="credithour" type="xsd:unsignedInt"/>
      <xsd:element name="section">
         < xsd:sequence>
            <xsd:element name="secnumber" type="xsd:unsignedInt"/>
            <xsd:element name="secyear" type="xsd:string"/>
            <xsd:element name="student">
               < xsd:sequence>
                  <xsd:element name="regnumber" type="xsd:unsignedInt"/>
                  <xsd:element name="stname" type="xsd:string"/>
                  <xsd:element name="stclasss" type="xsd:unsignedInt"/>
                  <xsd:element name="stgrades" type="xsd:unsignedInt"/>
               </xsd:sequence>
            </xsd:element>
         </xsd:sequence>
      </xsd:element>
   </xsd:sequence>
</xsd:element>
</xsd:element>
```

Figure 6 : XML Schema Document

References:

[1] Panos Vassiliadis, ALkis Simitsis, *Spiros Skiadopoulos. Conceptual Model for E TL Processes.* Proceeding of 5th ACM Int't workshop on Data Warehousing & OLAP. November 2002.

[2] Halena Galhards, Daniela Florescu, Dennis Shasha. *Declarative Data Cleaning: Language, Model & Algorithms.* Proceeding of the 27th VLDB Conference, Roma, Italy, 2001.

[3] Sunita Sarawagi. *Cleaning Methods in Data Warehousing.* KR School of Information Technology IIT, Bombay, December 1999.

[4] Rohit Ananthakrishna, Surajit Chaudhuri, Venkatesh Ganti. *Eliminating Fuzzy Duplicates in Data Warehousing.* Proceeding of the 28th VLDB Conference , Hong Kong,China, 2002.

[5] Halena Galhards, Daniela Florescu, Dennis Shasha. *An Extensive Framework for Data Cleansing.* In Proceeding of ICDE, 2000.

[6] Yan Zhu, Christof Borhovd, Alejandro P. Buchman. *Data Transformation for Warehousing Web Data.* Proceeding of 3rd Int'l workshop on advanced issues of E-Commerce & Web based Information Systems. June 2001.

[7] Peter Buneman, Susan Davidson, Gerd Hillebrand. *A Query Language & Optimization Techniques for Unstructured Data.* Proceeding of ACM Int'l Conference on Management of Data, June 1996.

[8] Matteo Golfarelli, Stefano Rizzi. *A Methodological Framework for Data Warehouse Design.* ACM 1st Int'l workshop on Data Warehousing & OLAP, November 1998.

[9] Jennifer Widom. *Reseach Problems in Data Warehousing.* Proceeding of ACM 4th Int'l Conference on Management of Data, 1995.

[10] Yingwei Cui, Jennifer. *Lineage Tracing for General Data Warehousing Transformation.* Proceeding of the 27th VLDB Conference, Roma, Italy, 2001.

[11] C. Shilakes, J. Tylman. *Enterprise Information Portals. Enterprise Software Team.* http://www.sagemaker.com/cpmpany/downloads/eip/indepth.pdf

[12] B. Inmon. *The Data Ware housing Budget.* DM Review Magazine, January 1997.

[13] Sophie Cluet, Pierangelo Velri, Dan Vodislav. *View in Large Scale XML Repository.* Proceeding of the 27th VLDB Conference, Roma, Italy, 2001.

[14] Vijayshankar Raman, Joseph M. Hellerstein. Potter Wheel: *A Interactive Data Cleaning System.* Proceeding of the 27th VLDB Conference, Roma, Italy, 2001.