ISSN:0975-9646



# A Novel Approach for Integrating Heterogeneous Database through XML

*V.Rajeswari*, Dr. Dharmishtan K. Varughese* **

* *Department of Information Technology, Karpagam College of Engineering, Coimbatore 32*

** *Department of Electronics and Communication Engg, Karpagam College of Engineering, Coimbatore –32*

***Abstract*** **- This paper illustrates an interactive mapping technique based on XML technology being tested in some possible scenarios and its adaptability verified in simplifying the integration and exchange of data between different database systems. This work is part of an effort in the development of a new technique for storing and mapping XML and different types of databases. The data from XML are compiled to MS-SQL Server, Oracle, MySQL, MS-Access and IBM-DB2.**

**Keywords – Heterogeneous databases, DOM, XML Parser, SAX.**

## I. INTRODUCTION

Database systems are the store house tools and backbones of the IT industry. In the early days the schemas for the different databases were developed without consideration of data integration with other systems [1]. They had different terminologies and structures. Since world information in the foreseeable future will continue to be saved in relational data stores because of scalability, reliability and performance reasons, it is now imperative to be able to map XML to relational databases. It is used as a common data format for cross-platform information exchange over the internet [3]. It separates presentation from data structure and content. But its rate of acceptance has been limited by the mismatch between XML and legacy relational databases [2]. So a mediator is required to map XML documents to legacy RDBMS.

Over the last few years, XML has become an indisputable standard both for data exchange and content management. XML has made an impression that XML databases will eventually replace more traditional RDBMS. Now the industry has started to move towards XML. As a consequence, many thousands of database and application developers are now facing the development challenge of converting XML data to a relational format which is a prerequisite for storing the vast amount of data in repositories. XML is frequently used for publishing as well as exchanging data.

## II. NEED FOR EXTRACTING DATA FROM HETEROGENEOUS DATA SOURCES

Data is typically stored in many different data storage systems[4]. Extracting data from all sources and merging the data into a single, consistent dataset is challenging. This situation can occur for a number of reasons.

Trend Analysis refers to the concept of collecting information and attempting to spot a pattern or trend [6] in the information. Trend Analysis [6] refers to techniques for extracting an underlying pattern of behavior in a time series which would otherwise be partly or nearly completely hidden by noise. A simple description of these techniques is trend estimation which can be undertaken within a formal regression analysis. The data of the organization are stored in the legacy databases. These data are not useful for the day-to-day operations. They are valuable for trend analysis that requires data collected over a long period of time.

A Data warehouse [7] is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process. A data warehouse provides a common data model for all data of interest regardless of the data's source. This makes it easier to report and analyze information from multiple data models. In a typical data warehousing application, the branches of an organization [7]may use different data storage technologies to store the operational data. The application package may need to extract data from different relational databases before it can merge the data.

## III. XML IN EXTRACTING DATA FROM HETEROGENEOUS DATA SOURCES

- XML can be used to describe and identify information accurately [10] and unambiguously, in a way that computers can be programmed to understand the information.
- In XML model the documents and fragments of documents are represented as ordered and node-labeled trees [10].
- XML allows documents which are all the same type to be created and handled consistently [10] and without structural errors, because it provides a standardized way of describing, controlling, or allowing/disallowing particular types of document structure.
- XML provides a robust and durable format for information storage and transmission. XML is robust because it is based on a proven standard and it can be tested and verified. XML is persistent because it uses plain-text file formats which will outlast proprietary binary ones.
- XML provides a common syntax for messaging systems for the exchange of information between applications. Previously, each messaging system had its own format and all were different, which made inter-system messaging unnecessarily messy, complex and expensive.

If everyone uses the same syntax it makes writing these systems much faster and more reliable.

- XML is a freeware [10]. XML information can be manipulated programmatically. So XML documents can be pieced together from disparate sources or taken apart and re-used in different ways. They can be converted into any other format with no loss of information.
- XML separates presentation from content. XML file contains the document information and identifies its structure. The formatting and other processing needs are identified separately in a style sheet or processing system. Several application domains like e-commerce and bibliographic references provide XML information on the web[17].

The unique strength of using XML as a software data format includes:

a. Simple syntax
b. Easy to generate and parse
c. Support for nesting
d. Nested elements allow programs to represent complex structures easily
e. Easy to debug
f. Language and Platform Independent
g. Openness
h. Extensibility
i. Self Description
j. Rapid adoption by industry
k. Contains machine readable Context Information[15]
l. XML and Unicode guarantee that the data files will be portable across virtually every popular computer architecture.

## IV. PROBLEMS IN HETEROGENEOUS DATABASE INTEGRATION

Heterogeneous database integration systems are computational models and software implementations. They provide a single, uniform query interface to data that are stored and managed in multiple, heterogeneous data sources [12][24].The goal of such systems is to provide database transparency to users as if the data were not distributed and all of the data sources were of the same type. In spite of standardization efforts, the database heterogeneity still remains. This occurs mostly in web applications. Some of the main challenges are:

*a) Query Processing Models*

The core challenge of heterogeneous database integration is that different data sources have different query models [12]. A query model is the model of data storage and information retrieval. The user of the database must know the order in which the data is to be retrieved from the database. A query model consists of the data model, database schema, query language and data format. For example the data may be stored variously stored in flat files, XML files, binary files, spreadsheets and in relational object-relational and object-oriented databases[14].

*b) Independent Data Sources*

Generally data sources are autonomous. This means that developers of the heterogeneous database do not have control over the data sources to be integrated. Each data source is free to modify its data and schema and to restrict access to it.

*c) Semantic Heterogeneity*

Similar data are often represented differently in different data sources. This representational heterogeneity consists of structural, naming, semantic and content differences [19]. Structural differences refer to schema differences. Naming or syntactic [11] differences occur when semantically equivalent objects are named differently in different data sources. Semantic differences occur when names of objects in different data sources are similar or the same, but their meanings differ. Content differences refer to differences in data between the data sources.

Semantic conflict exists when the communicating parties use different representations or interpretations of the information that is being communicated. Interpersonal communication can require an explicit translation process. This process is called as semantic reconciliation [11].

Just as semantic reconciliation must occur in interpersonal communication, it must also occur when different information systems attempt to communicate.

Databases can also be heterogeneous on many levels. They can have different data models or different conceptual models. It means that they have different meanings and different representations of the same reality. They may also have different naming conventions [11] and different ways to organize the information. It is imperative to understand that these lead to conflicts.

The following are the types of conflicts:

- Value-to-Value conflicts
- Value-to-Attribute conflicts
- Value-to-Table conflicts
- Attribute-to-Attribute conflicts
- Attribute-to-table conflicts
- Table-to-Table conflicts

*d) Technical Heterogeneities:* These challenges occur due to differences in hardware platforms, operating systems, access protocols, transport formats and programming languages between the data sources.

## V. NEED TO IMPORT XML DOCUMENTS TO RELATIONAL DATABASES

A number of reasons can be cited warranting the need to move XML to a relational data store. Businesses across industries are adopting XML to share data between applications (A2A) and organizations (B2B). XML is a natural and persistent way for heterogeneous database systems to share data across networks. In addition, it is rapidly being adopted by software companies as a standard means of sending and receiving data. In fact, XML messaging is progressively starting to replace the proprietary EDI format

for business-to business transactions. XML's self-describing nature [8] makes it easy to exchange transactional data between business partners and incompatible systems. Therefore, it largely simplifies B2B [12] communications and is seen as a cost-effective and open-standard replacement for EDI.

XML database's biggest competitors are relational databases such as Oracle or SQL Server. They have started to interface directly with XML data. But the RDBMS-XML interfaces [13] differ from vendor to vendor and are not portable. The difference between these two data storage formats makes it very difficult to map one to the other. The significant development and customization is necessary to import XML documents to relational databases.

## VI. HETEROGENEOUS SYSTEM STRUCTURE

The data warehouse creation module is regarded as user of the heterogeneous database system. The user poses a global query on the integrated system. The global site decomposes the global query into sub-queries to request each participant to return the data in XML format [1]. This can be easily achieved in contemporary commercial database products. Besides, the DBA in each site should prepare some XSLT format files, together with some necessary template files, in advance to transform local data into the global schema format [2]. Finally, the transformed data are integrated into a consolidated view with the global query applied on it to return global data to the user. The integration process utilizes the semantic knowledge of all participate local schemas. This should be prepared or discovered before integration.

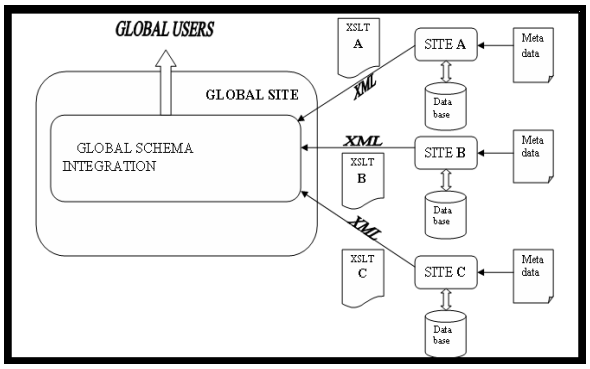


fig 1 : Heterogeneous Database Architecture

In data warehousing, data from each source is extracted, merged and stored in a centralized repository [15]. The warehouse is a database with a global schema that combines the schemas of the sources. Queries on the system are evaluated at the warehouse without accessing the original sources. Client updates to the warehouse are usually not allowed. Since they are not propagated to the original sources and would make the warehouse inconsistent with the sources. Instead, the warehouse is updated from the data in the original sources. There are multiple policies for updating the warehouse from the sources.

Database integration is required for data warehousing. It relies less on the network and allows for improved query performance and optimization, since queries are processed locally in the warehouse.

## VII. MEDIATION

In mediation a technique called a "mediator" accepts a query from the client, determines the sources needed to answer the query and decomposes the query into sub queries. The sub queries are translated to the source-specific query language via modules called "wrappers." The results from the sources are translated back into the common query language by the wrappers. Finally, the mediator obtains results from the wrappers, combines them and returns the final answer to the client.

Mediation can be query-centric known as global-as-view or source-centric known as local-as-view. In query-centric mediation, users pose queries on the global views exported by the mediator. The mediator uses global views to expand user queries to queries on source data.

In source-centric mediation, global predicates are used to construct source views and to pose user queries. The mediator uses source views to answer user queries. The main advantage of mediation is that it reduces the maintenance required when data sources are modified or when new ones emerge.

## VIII. XML DATABASE TYPES

There are two categories to consider when deciding which type of XML database fits a particular application:

- Data-centric[9]: Products that actually store the data or content in non-XML format
- Document-centric: Products that store complete XML documents in relational tables or on disk in file structures

Data-centric databases store data separate from the XML schema, usually just transforming the original content into relational tables. These products are referred to as XML-enabled databases.

If an XML document is needed, the data stored in relational tables can be queried and an XML document created. Most major relational databases ORACLE and SQL Server fall into this category.

Document-centric databases store the entire XML document in a relational, text, or proprietary format. These are called native XML databases.

## IX. INTEGRATION OF QUERY , SELECTION AND TRANSFORMATION PROCESS

The larger objective is to switch from a database-centric approach of application development to one that is based entirely on the XML paradigm [6]. One side effect of this paradigm shift is the possibility of using standardized XML processors – including XQuery, XPath and XSLT – to further query, filter and transform the XML views obtained from relational data by means of mapping techniques. One

obvious difficulty of running XML processors on mapped XML views is that XML views are not materialized unless cached in memory, stream or file. This may become a blocking issue if the XML view obtained from the mapping process is large. Either the cached view is too large for memory or disc space or the time needed to post-process the XML view by an XML processor is unacceptably long.

To overcome these problems the following strategies are used.

- Develop mapping aware XML processor. Such processors understand mapping definitions and translate XML processor statement into native SQL statements.

- The standard XML processors from software vendors or open source are used. The first objective is to reduce the size of the XML record set generated by mapping process. To reduce the size of the XML record set, they can perform an initial parsing of the XML processor query statement to extract only the relevant predicate information to append to the SQL generated by the mapping process.

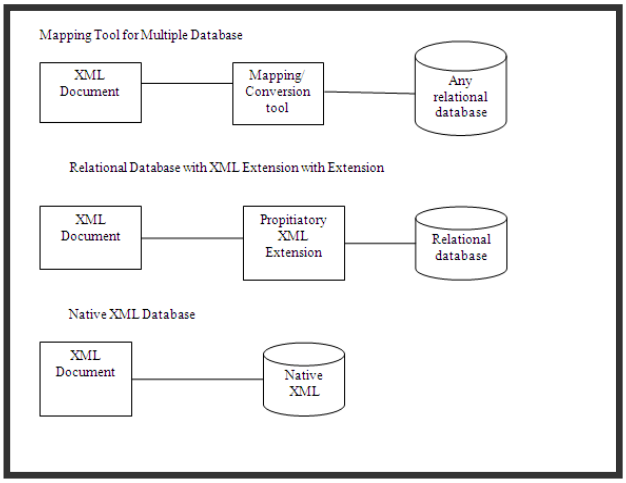


Fig. 2. Integration of Query Selection

In accordance with the query selection process illustrated in Figure.2, the new technique discussed, primarily detects the type of the database used and decides the course of action.. Then according to the document-centric approach of the XML database type, the technique stores the complete XML document into relational tables.

## X. PROGRAMMING INTERFACE

A variety of APIs for accessing XML have been developed and used, and some have been standardized. Existing APIs for XML processing tend to fall into these categories:

- Stream-oriented APIs accessible from a programming language, for example SAX and StAX.
- Tree-traversal APIs accessible from a programming language, for example DOM and XOM.
- XML data binding, which provides an automated translation between an XML document and programming-language objects.
- Declarative transformation languages such as XSLT and XQuery.

Stream-oriented facilities require less memory for certain tasks which are based on a linear traversal of an XML document. They are faster and simpler than other alternatives.

Tree-traversal and data-binding APIs typically require the use of more memory, But they are found to be more convenient for use by programmers. Some of them may include declarative retrieval of document components through decides the cour the use of XPath expressions.

## XI. SAX INTERFACE

SAX [14] is a lexical, event-driven interface in which a document is read serially and its contents are reported as "callbacks" to various methods on a handler object of the user's design. SAX is a common front-end for XML parsers, like the JDBC for database access. SAX is widely used by open-source projects like Apache and by corporate users like Sun, IBM, Oracle and Microsoft.

The SAX parser is created by the javax.xml.parsers.SAXParserFactory. The SAX parser does not create an in-memory representation of the XML document and so it is faster and uses less memory. Instead, the SAX parser informs clients of the XML document structure, by invoking callbacks, which is by invoking methods on an org.xml.sax.helpers.DefaultHandler instance provided to the parser.

The DefaultHandler class implements the ContentHandler, the ErrorHandler, the DTDHandler and the EntityResolver interfaces. The clients will call the methods definesd in the ContentHandler interface. Those methods are called when the SAX parser encounters the corresponding elements in the XML document. The most important methods in this interface are:

- startElement() and endElement(): These methods are called at the start and end of a document element.
- characters() : This method is called with the text data contents contained between the start and end tags of an XML document element.

Clients provide a subclass of the DefaultHandler that overrides these methods and processes the data. This involves storing the data into a database or writing it out to a stream. During parsing, the parser may need to access external documents.

## XII. XML PROCESSING WITH SAX

A parser which implements SAX, functions as a stream parser, with an event-driven API. The user defines a number of callback methods that will be called when events occur during parsing. The SAX events include:

- XML Text nodes
- XML Element nodes

- XML Processing Instructions
- XML Comments

Events are fired when each of these XML features are encountered, and again when the end of them is encountered. XML attributes are provided as part of the data passed to element events. SAX parsing is unidirectional; previously parsed data cannot be re-read without starting the parsing operation again.

*Example*

```
<?xml version="1.0" encoding="UTF-8"?>
<EMP>
<EMPNO> IT1000 </EMPNO>
<ENAME> JAMES </ENAME>
<JOB>CEO</JOB>
<HIREDATE>21-SEP-2000</HIREDATE>
</EMP>
```

fig 4 : XML Document

This XML document [25] [16], when passed through a SAX parser, will generate a sequence of events like the following:

- XML Processing Instruction, named *xml*, with attributes *version* equal to "1.0" and *encoding* equal to "UTF-8"
- XML Element start, named *RootElement.*
- XML Element start, named *EMPNO*
- XML Text node, with data equal to "IT1000" XML Element end, named *EMPNO*
- XML Element start, named *ENAME*
- XML Text node, with data equal to "JAMES"
- XML Element end, named *ENAME.*
- XML Element start, named *JOB*
- XML Text node, with data equal to "COE"
- XML Element end, named *JOB.*
- XML Element start, named *HIREDATE*
- XML Text node, with data equal to "21-SEP-2000"
- XML Element end, named *HIREDATE.*
- XML Element end, named *EMP*

The SAX specification deliberately states that a given section of text may be reported as multiple sequential text events. Thus in the example above, a SAX parser may generate a different series of events, part of which might include:

- XML Element start, RootElement *named EMP*
- XML Element start, named *EMPNO,EMPNAME,JOB,HIREDATE*
- XML Text node, with data equal to "IT001,JAMES,COE,21-SEP-2001 "
- XML Text node, with data equal to "Text"
- XML Element end, named EMP.

## XIII. TRANSFERRING DATA BETWEEN XML DOCUMENTS AND RELATIONAL DATABASES

The work that has been carried out with reference to the objective of integrating heterogeneous data bases is closely dependantant on various technologies that go into the creation of information systems.

Technologies assoicated with the creation of information systems are:

- Object-oriented reperesentation
- Relational databases
- Extensible Markup Language(XML)

Sharing of Objects and XML is reduced into following tasks

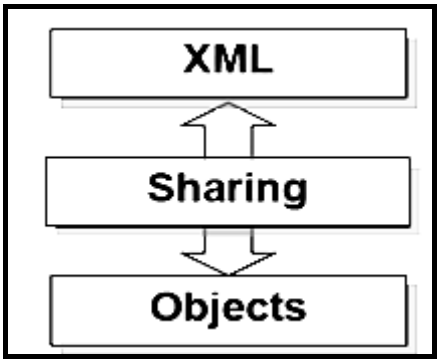


fig 5 : Sharing of Objects and XML

a) Representation of Objects as XML (marshalling)
b) Transformation of XML to structures of Objects (Unmarshalling)

The proposed technique , part of a new framework is the core of a middleware, connected as shown in fig.6. It has the functional role in :

- Storing the XML documents in the relational databases
- Loading the XML documents from the relational database

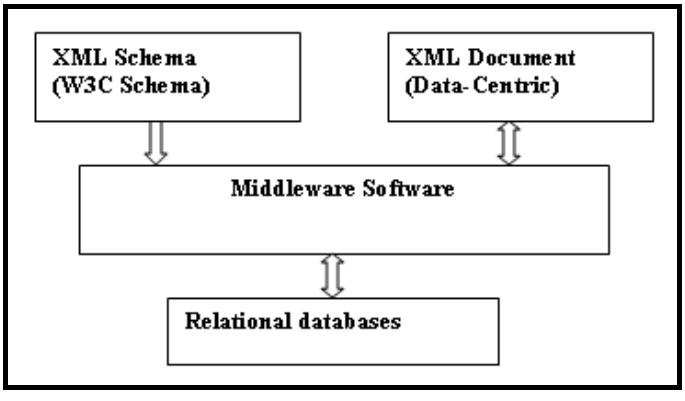


fig 6 : Relationship between XML and Relational Database

ISSN:0975-9646



# A Novel Approach for Integrating Heterogeneous Database through XML

*V.Rajeswari*, Dr. Dharmishtan K. Varughese* **

* *Department of Information Technology, Karpagam College of Engineering, Coimbatore 32*

** *Department of Electronics and Communication Engg, Karpagam College of Engineering, Coimbatore –32*

***Abstract*** **- This paper illustrates an interactive mapping technique based on XML technology being tested in some possible scenarios and its adaptability verified in simplifying the integration and exchange of data between different database systems. This work is part of an effort in the development of a new technique for storing and mapping XML and different types of databases. The data from XML are compiled to MS-SQL Server, Oracle, MySQL, MS-Access and IBM-DB2.**

**Keywords – Heterogeneous databases, DOM, XML Parser, SAX.**

## I. INTRODUCTION

Database systems are the store house tools and backbones of the IT industry. In the early days the schemas for the different databases were developed without consideration of data integration with other systems [1]. They had different terminologies and structures. Since world information in the foreseeable future will continue to be saved in relational data stores because of scalability, reliability and performance reasons, it is now imperative to be able to map XML to relational databases. It is used as a common data format for cross-platform information exchange over the internet [3]. It separates presentation from data structure and content. But its rate of acceptance has been limited by the mismatch between XML and legacy relational databases [2]. So a mediator is required to map XML documents to legacy RDBMS.

Over the last few years, XML has become an indisputable standard both for data exchange and content management. XML has made an impression that XML databases will eventually replace more traditional RDBMS. Now the industry has started to move towards XML. As a consequence, many thousands of database and application developers are now facing the development challenge of converting XML data to a relational format which is a prerequisite for storing the vast amount of data in repositories. XML is frequently used for publishing as well as exchanging data.

## II. NEED FOR EXTRACTING DATA FROM HETEROGENEOUS DATA SOURCES

Data is typically stored in many different data storage systems[4]. Extracting data from all sources and merging the data into a single, consistent dataset is challenging. This situation can occur for a number of reasons.

Trend Analysis refers to the concept of collecting information and attempting to spot a pattern or trend [6] in the information. Trend Analysis [6] refers to techniques for extracting an underlying pattern of behavior in a time series which would otherwise be partly or nearly completely hidden by noise. A simple description of these techniques is trend estimation which can be undertaken within a formal regression analysis. The data of the organization are stored in the legacy databases. These data are not useful for the day-to-day operations. They are valuable for trend analysis that requires data collected over a long period of time.

A Data warehouse [7] is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process. A data warehouse provides a common data model for all data of interest regardless of the data's source. This makes it easier to report and analyze information from multiple data models. In a typical data warehousing application, the branches of an organization [7]may use different data storage technologies to store the operational data. The application package may need to extract data from different relational databases before it can merge the data.

## III. XML IN EXTRACTING DATA FROM HETEROGENEOUS DATA SOURCES

- XML can be used to describe and identify information accurately [10] and unambiguously, in a way that computers can be programmed to understand the information.
- In XML model the documents and fragments of documents are represented as ordered and node-labeled trees [10].
- XML allows documents which are all the same type to be created and handled consistently [10] and without structural errors, because it provides a standardized way of describing, controlling, or allowing/disallowing particular types of document structure.
- XML provides a robust and durable format for information storage and transmission. XML is robust because it is based on a proven standard and it can be tested and verified. XML is persistent because it uses plain-text file formats which will outlast proprietary binary ones.
- XML provides a common syntax for messaging systems for the exchange of information between applications. Previously, each messaging system had its own format and all were different, which made inter-system messaging unnecessarily messy, complex and expensive.

| | empno | EMPNAME | JOB | HIREDATE |
|---|---|---|---|---|
| | 1 | JACK | PROJECT LEADER | 21-SEP-01 |
| | 2 | JAMES | TEAM LEADER | 14-APR-99 |
| | 3 | PAUL | SYSTEM ADMIN | 14-SEP-01 |
| | 4 | FREDRICK | CEO | 23-MAR-94 |
| | 5 | SIMON | CC HEAD | 2-MAY-01 |

fig 10: Records inserted into ACCESS from XML File

| EMPNO | ENAME | JOB | HIREDATE |
|---|---|---|---|
| 1 | JACK | PROJECT LEADER | 21-SEP-01 |
| 2 | JAMES | TEAM LEADER | 14-APR-99 |
| 3 | PAUL | SYSTEM ADMIN | 14-SEP-01 |
| 4 | FREDRICK | CEO | 23-MAR-94 |
| 5 | SIMON | CC HEAD | 2-MAY-01 |

5 record(s) selected.

fig 11: Records inserted into MYSQL from XML File

| EMPNO | ENAME | JOB | HIREDATE |
|---|---|---|---|
| 1 | JACK | PROJECT LEADER | 21-SEP-01 |
| 2 | JAMES | TEAM LEADER | 14-APR-99 |
| 3 | PAUL | SYSTEM ADMIN | 14-SEP-01 |
| 4 | FREDRICK | CEO | 23-MAR-94 |
| 5 | SIMON | CC HEAD | 2-MAY-01 |

5 record(s) selected.

fig 12: Records inserted into DB2 from XML File

During the process of insertion into the database, the processing time of each database management system is measured.

| S.No | DBMS | Processing Time |
|---|---|---|
| 1 | ORACLE 9i | 10 |
| 2 | DB2 Express C | 24 |
| 3 | SQL Server | 1434 |
| 4 | MYSQL | 535 |
| 5 | MS-ACCESS | 5545 |

fig 13: Processing Time for different databases.

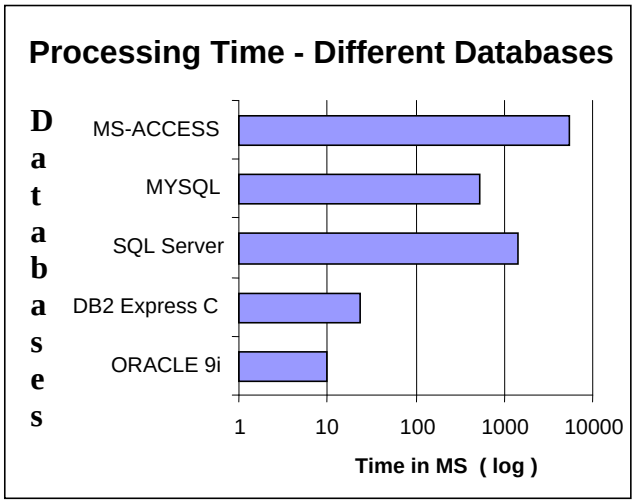


fig 14: Processing Time – Graphical Representation

## XVI. CONCLUSION

The creation of an integrated interface over a given set of existing heterogeneous databases is a challenge faced by many database administrators today. This paper has identified some of the problems that arise during the schema integration process. A methodology has been proposed for the creation of an integrated schema from a given set of local database schema. A Parser parses the file and the processing time is calculated. The processing speed is calculated for different types of databases and presented for evaluation.

If the application requires moving data between enterprises, XML is a good solution. XML sends data across the Internet and through firewalls by using the standard HTTP protocol. XML is also a good choice if application needs to move data between hardware or software platforms.

## REFERENCES

[1] Frank S.C. Tseng, *Heterogeneous database Integration Using XML.*

[2] Frank S.C. *XML based heterogeneous Database Integration for Data warehouse Creation.*

[3] Yanxinwang, Kuiheyang *"Research and Realization of XML Based Heterogeneous Databases Integration"*

[4] Wei- Jung Shiang, Minng-Ying Ho, *"An Interactive Tool Based on XML Technology for Data Exchange between Heterogeneous ERP systems"*, Journal of CIIE, Volume 22, No.4, pp.273-278(2005).

[5] Guardalven "Integrating *XML and Relational Database Technologies*" 8/12/04.

[6] Data Warehouses and OLAP: Concepts, Architectures and Solutions
By Robert Wrembel, Christian Koncilia
Category: OLAP
Published Date: 2006-10-30

[7] Shankar Pal,Istvan Cseri,Oliver Seeliger,Michael Rys,Gideon Schaller, Wei Yu, Dragan Tomic,Adrian Baras,Brandon Berg, Denis Churin, Eugene Kogan, *XQuery implementation in a Relational Database*

*System*,Proceedings of the 31st VLDB conference, Trondheim,Norway,2005.

[8] Journal of Organizational Computing 5(2),167-193, (1995), A Classification of Semantic Conflicts in Heterogeneous Database Systems Channah E Naiman, Aris M. Ouksel, *University of Illinois at Chicago*

[9] http://www.ncbi.nlm.gov/sites/ppmc/articles/ PMC2675489

[10] Felipe Victolla Silveira and Carlos A.Heuser, UFRGS, Brazil, *A Two Layered Approach for Querying Integrated XML Sources.*

[11] Data Warehouses and OLAP: Concepts, Architectures and Solutions
By Robert Wrembel, Christian Koncilia
Category: OLAP
Published Date: 2006-10-30

AUTHORS PROFILE

Ms. V. Rajeswari is Assistant Professor in the Department of Information Technology, Karpagam College of Engineering, Anna University of Technology, Coimbatore. Her areas of specialisation and research interest are J2EE and Database Management Systems.

Dr. Dharmishtan K. Varughese is Professor in the department of Electronics and Communication Engg, Karpagam College of Engineering, Anna University of Technology, Coimbatore. His areas of research specialistion are Database Management Systems, Micro Strip Antennae and Antenna and Wave propagation.