An Approach for Generic Information Query Retrieval in Web2.0

K Thippeswamy **J Hanumanthappa** **D H Manjaiah**

Abstract

With the growing amount of information in various domains, retrieval and analysis is the most frequently used operation. Various institutions/organizations generate valuable information in various domains which is queried and analyzed by users for various purposes. Most of the applications performing these tasks are predominantly database driven and tightly coupled with the system. This limits the possibilities of seamless database integration with other sources of knowledge and also their ability to adopt to changes in the information structures. We describe a generic approach comprising a loosely coupled system with an ability to perform complex querying, analysis and seamless integration with other systems, both off line and over Internet.

Keywords: SOA, Web Services, SOAP, Web 2.0

1. Introduction

Data analysis and mining is a major task in all sectors. Once information is generated, the major task is to integrate and exchange with other organization that use this information with their application for data analysis. Traditional approach consists of physically moving the information to other location or to have a tightly coupled system to use it. It becomes very difficult to maintain such system because of the ever changing information resources. Also in the era of WEB2.0 there is an increasing need of information exchange across Platform, across location on real time basis.

1.1 Existing System Problems

- Application to application communication is tightly coupled and requires constants maintenance due to tightly coupled behaviors.
- Traditional architecture lacks the ability to be reused and interact with new software.
- In the traditional system information exchange and integration becomes a expensive task.

1.2 Proposed Systems

- Creation of a loosely coupled information retrieval system.
- Approach to decompose the application in to services which can operate independently and also consume each other.
- Client program to access the generic framework services to do perform data analysis with out being dependent on the internal system.
- Approach to have application to application data communication by different means.
- Approach to develop a generic client module to communicate with the services. Fig 2 shows the Proposed System.
- Traditional client server architecture for information retrieval and analysis lacks the ability to expand and adapt to changes.

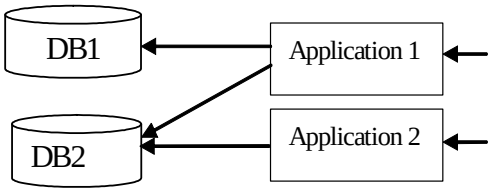


Figure 1. Existing System

2. Methodologies

The system can be decomposed to following modules which need to be worked upon to create a loosely coupled system.

- Model to represent the set in put parameter
- Model to create services with an aim to share common data framework.
- Model to have a loosely coupled system to maintain services (ESB).
- Generic query retrieval system which is loosely coupled and configurable and independent of data resources.

2.1 Model to represent the set in put parameter

Input search query can either in a delimiter separated form or can be in XML form. Details are given below.

- Delimiter separated format
- % COLi = [cond]val1[::]@@@[cond]val2[::]… i =0 to n
- The xml format is shown in fig 3.
- condition refers to condition like "!=","<" ,">" etc
- Match type can be strict where search should go to all the tables or can be loose where search continues if to next table if no match found in current table
- Group by and group to column are used to showing stats for grouping.

2.2 Model to create services with an aim to share common data framework.

In principle with SOA approach we have developed web services modules which can communicate with the clients and exchange information. We have used Apache axis2 engine for developing the web services. Two type of service model were developed which can be used by the clients programs. They are AXIOM and ADB.

AXIOM (Axis2 Object Model) Method.

Since xml is the universally accepted method of information exchange apache axis2 provides a XML object model for efficient SOAP messaging. Instead of the xml Axiom data object becomes the method of data transfer. It supports a novel "pull-through" model which allows one to turn off the tree building and directly access the underlying pull event stream. It also has built in support for XML Optimized Packaging (XOP) and MTOM, the combination of which allows XML to carry binary data efficiently and in a transparent manner. Since the nature of message transfer will be both **in and out** we have to define appropriate message receiver in the **Service.xml** file.

After receiving the Input AXIOM the web services will process the AXIOM and return the appropriate xml file as an AXIOM model.

ADB (Axis2 Data Binding) Method.

This model is the most advance model which provides the application to send and receive java object instead of the Axiom which they can directly

use in the application. The ADB framework provides a linking code from the WSDL file which can be directly used by the applications. From the WSDL file skeleton code can be generated where later we can put the business logic from the existing java code. Connecting code is created by the wsdl4java library. Based on the WSDL the wsdl2java API creates the connecting code which is called the stub code also. Following block Diagram shows the basic information flow of the ADB server model.

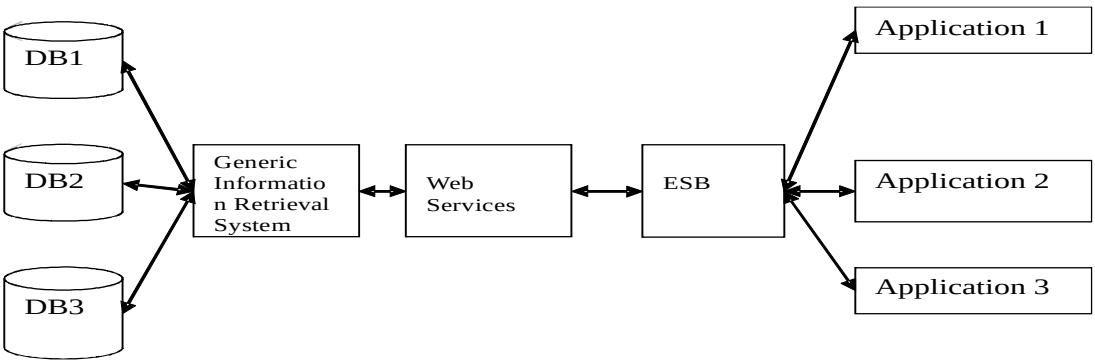


Figure 2. Information Exchange and Integration using SOA

2.3 Model to have a loosely coupled system to maintain services (ESB).

ESB (Enterprise Service Bus) system provides the infrastructure for implementing SOA architecture's acts as a mediation layer between the web services and the client application. Apache synapse is used as the ESB layer because of flexibility in maintaining the configuration files which are in xml format. Any change in the service configuration file can be done without actually restarting the servers. Since we have to models of web services so we need to have Context based routing mechanism by means of which synapse routes the incoming message to the appropriate web services.

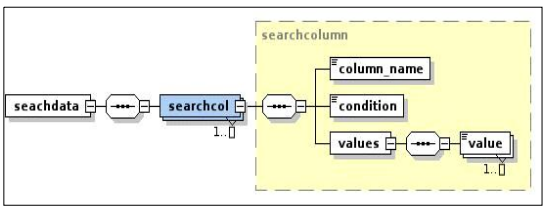


Figure 3. Xml representation of input query

Typical input AXIOM can look like figure 4

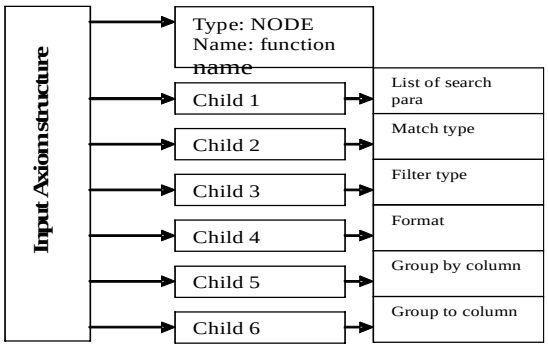


Figure 4: AXIOM Object Model

- We also need to have a proxy layer by means of which we can hide the actual location of different web services and give only one endpoint URL to the user.
- Following is a synapse configuration file form CBR (context based routing). Here the message is routed to different web services based on the message pattern.

- So the web services system looks like below with the ESB layer.

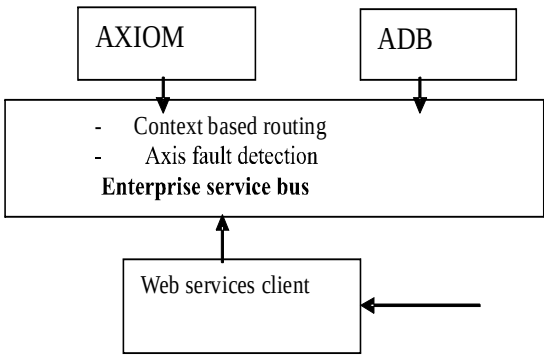


Figure 5: Web Service System

2.4 Generic Information retrieval system

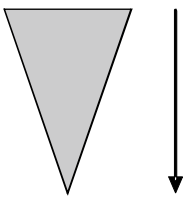
1. Model to represent the underlying relational database in a generic form.
2. Model to generate queries based on input conditions and querying, filtering, grouping across databases based on the input query and dispatch the output.

2.5 Relational database representation in XML format

XML format was preferred over the traditional delimiter separated file because of the ability of xml to show the hierarchical relations in an effective manner and also due to the flexibility to make changes without affecting the existing structure. An XML schema was design to represent the tables in a database and also the relations between the tables. Also we have tried to address the issue of heterogeneity in terms of column naming etc. Following block diagram shows the design of the schema searchcolumn refers to the columns of the table available for searching, displaycoulmn refers to the column data to be displayed in case of table specific data Display. The synonym list section contains more than one synonym block which is for displaying the relations Between tables. Each synonym block has a primarycolumn element which refers to the columns which are related to other tables, secondcomlumn element are column name which are present in other tables but with a different nomenclature or in other words the related column to the primarycolumn in other tables. Same is depicted in figure 6

2. 6 Model to perform querying and filtering

Querying and filtering is design from the perspective of narrowing down on the most specific record based on the series input data. After Input condition is processed ,for each search condition corresponding table details are retrieved and stored .Then for each table details queries are formed and data is retrieved. Each table query also has the output of previous query as a part of the condition thus relating to the previous table. There are two methods for filtering, they are 1.strict, 2.loose. Basic flow of query is given below.



We start with the first input Second query will have output of the previous stage as condition in addition to the condition provided in the input query.

In the strict method the process of querying stops if no result id found at some stages, on the other hand in the loose method querying continues in the next table with output of previous stage where match was found as a condition in addition with the condition provided in the input query.

Algorithm

1. Repeat step 3 and 4
2. Search column for table.
3. If table found, store table name, search column, search value and search condition
4. Initialize the result key value pair
5. For each stored table information in step4
   a) Check if result key value pair has any entry which is equal to the primary column of the current table
   b) If yes form the appropriate query
   c) Execute the query
   d) If strict option selected then
   e) Get the distinct value of primary column and store as key value pair replacing earlier keys if present with the new value.
   f) If no value found then empty the result key-value pair

   Else

Get the distinct value of primary column and store as key value pair replacing earlier keys if present with the new values.

6. After query of primary keys are obtained, data is retrieved

a) If normal output

If group by is enabled then

If match is yes

For each group by primary id

For each match table make query

Execute the query and store the data in appropriate objects

Else

If match is all

For each group by group by primary id

For each match table make query

Execute the query store the data in appropriate object.

Else

For each table make query

Execute the query store the data in appropriate object.

Else

Get the global xml from each primary key

2.7 Model to perform filtering and grouping

Grouping is done with a prospective of classifying the output by a column which is referred as group by column. This in important from statistical analysis, bar graph etc .Also it is very significant in domains like life science, finance. We have designed the model to perform a grouping operation between two column values not only in the same table but any tables in the system. After the filtering as described in the above step we have a series of key value pair as output. The details of group by and group column are searched. Then for each value of group by column value subset of the output key-value pair is taken and then the corresponding group column is queried in the respective table of group column. So each group we obtain the data of group column which can be taken for next level of grouping.

a) Method1

Algorithm

1. Get cluster column
   Search for table info
   Get the output from step 6.
2. If cluster is done with specific values of cluster column then make a count and

group query on the group cluster table with output from step 6 and specified value of Custer column.

Else

Maker a count and group query on the group cluster table with output from step 7 of querying algorithm.

b) Method2

1. Get cluster column x with value if any

2. Get cluster column y with value if any
   Search table info of cluster x
   a. If cluster is to done with specific values of cluster column x then search specific values in respective table
      else
      Get distinct value from specific table
   b. For each value of distinct cluster column x
   c. Create query from output from step 7 of querying algorithm.
3. Execute query for each value of cluster x column and Store the out as key value pair where key is the value of cluster x column and value is again a key value pair of output of query
4. Search for table information of cluster column y
   For each key value pair obtained in step 4
   a. Create and execute query
   b. Get the count for each value of cluster x column
5. Generate the XML output

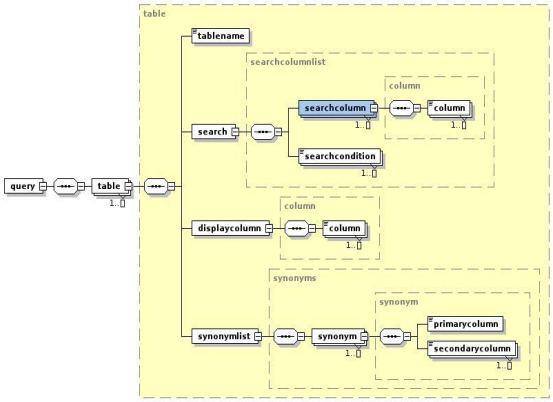


Figure 6. Xml schema showing xml representation of relational database

2.8 Implementation

Apache axis2 is used as the web service platform along with Apache Tomcat .The web services model is described in the Figure 5. The block diagram of SOA approach for information retrieval using web services system is shown in figure 7 and it is implemented using java platform. The xml parsing and xslt transformation is implemented using Apache Xerec and Xalan API. Object xml mapping was implemented with Castor API. The query interpreter processes the input search data. The SQL query creator perform the process of finding table info of the search columns and forming SQL queries and then the SQL query executer performs execution and storing required key value pair as discussed in the section in the discussion 2.4. After the primary key retrieval, data is retrieved based on the method selected .If no global xml data is presents then data is stored in data objects. Based on the client's request the data is converted to xml using Object xml mapping concepts using Castor API or converted to HTML using XSLT transformation on the output of above step or converted to the AXIOM model and transmitted else data object itself is transmitted .

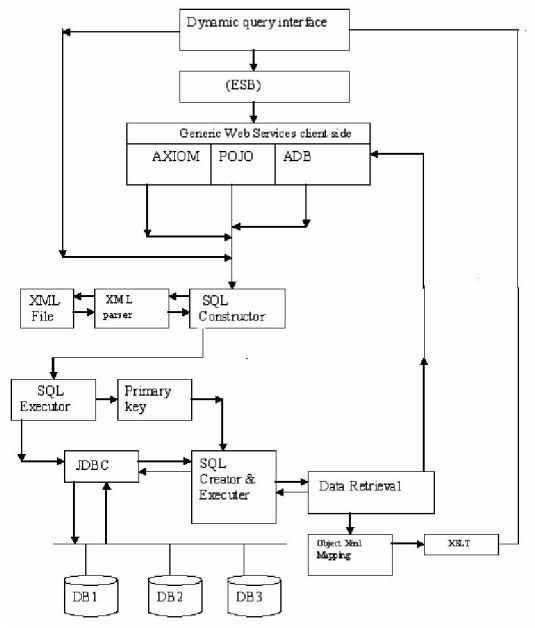


Figure 7: Block Diagram of Web Services System

3. Conclusion

In this paper we presented an approach and implementation of a generic querying retrieval system which can used for querying and analysis with each module being loosely coupled with other. Following the summery of results achieved

1. Context independent and configurable module design.
2. Service oriented model design where each module behaves independent of each other.

References

1. **Jerry R.Hobbs,** Artificial Intelligence Center, SRI International, Menlo Park, CA 94025, A Generic Information Extraction System.

2. **Ch.Dreier,** Generic Data Access in XML-Based LighweightWorkflow Management System, In German., Master's thesis, University of Klagenfurt, 2005.

3. **Eyal Oren, Armin Haller, Manfrad Hauswith, Benjamin Heitmann and Stefan Decker**< National University of Ireland, Gateway .Cedric Mesnage, University of Lugano. A Flexible Integration Framework for Semantic Web 2.0 Applications.

4. **Ch. Dreier,**Generic Data Access in XML-Based LighweightWorkflow Management System,In German., Master's thesis, University of Klagenfurt, 2005.

5. **Amit Sheth and James Larson,** "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", ACM Computing Surveys 22(3): 183-236 (1990)

Acknowledgement

I would like to thank the Department of Information Science & Engg. R.L.Jalappa Institute of Technology, Department of Computer Science at University of Mysore & Mangalore University for providing support to conduct this research work.

About Authors

Mr. K Thippeswamy, Assistant Professor & HOD, Dept. oF Information Science & Engineering, R .L. Jalappa Institut of Technology. Kodiehalli, Doddaballapura, Bangalore

Mr. J Hanumanthappa, Lecturer in Department of Studies in Computer Science ,Manasagangothri, University of Mysore.

Dr. D H Manjaiah, Reader and Chairman of BoS in both UG/PG in the Computer Science at Dept. of Computer Science, Mangalore University, Mangalore.