THE INTEGRATION OF HETROGENEOUS DATA SOURCES FOR QUALITY BASED DYNAMIC SOURCE INTEGRATION SYSTEM

Rahul Patel*

Abstract:

Data integration is a challenging domain for combing data from different sources. This provides users a unified view of data. Data integration system constitutes a major importance in current real time application and characterized by some issues related from speculative conclusion of view. Integration of multiple heterogeneous data sources into real time application is a time consuming and costly process. Advance technologies and standard make integration well managed from data engineering perspectives followed by many limitations. These limitation projects a different area of research to provide solution to these challenges. In our research we cover an overview of some theoretical issues that are relevant to data integration followed by proposed ablution dealing with inconsistent data source, provide reasoning on queries. Our proposed methodology will include schema and data integration, query processing in distributed data environment followed by query optimization to overcome data inconsistencies. The need of integration of heterogeneous data sources has influenced the area of research in the past few years.

Keywords: Data integration, Heterogeneous Data source , Query Transformation, Data Model, Metadata Repository.

* M. Tech Student, Medicaps Institute of Technology & Management College Indore.

1.Introduction

Search engines on the World Wide Web have to learn to deal with the rapid increase in volume and diversity of online information and continuous changes in the content and location of data sources. There is on important property which charities search engines and which makes them so successful, namely the simplicity of formulating queries. Queries are mostly specified by simply typing in keyword and the search engine finds the documents that may fulfill the given information need. However, the price for this simplicity in querying is the quality of answers. World Wide Web search tolls often return too many results and, at the same time, results which are relevant to users are not returned. Modern database management system also does a good job of managing and providing access to a large volume of data. The data is stored together with its structure, and queries can be formulated using the structure and a query language. The main advantage is the high quality of answers. In other words, the price for relevant results has to be played by users through the lack of simple formulation of queries. The need to integrate heterogeneous data sources has influenced the general direction that research has taken in the past few years. The focus of this research is to design a global information system which integrates structurally heterogeneous data sources, i.e. source which manages structured, semi-structured and unstructured data, from the point of view of querying.

Data integration is becoming more and more significant with the development of the data network. At present, the most popular manner of data integration is integration of heterogeneous data sources. In the model of integration of heterogeneous data sources e, data is abstracted from the source data and pretreated to satisfy the target data, and then is updated to target data. This method is always applied in the index database systems but it has following defects:

1. Data Source integration is unilateralism, the target data is just the receiver but not the provider of data;

2. The high delay of the data synchronization exists, so the facility of the data source is poor.

3. The procedure of the pretreatment is difficult. We have to write a lot of codes to realize the data conversion.

Some data integration systems adopt virtual view to integrate the source data, this method has good performance when the source data is unite or compatible. Furthermore, the present databases are huge; complete data synchronization is unpractical for its large delay. In order to mitigate these above problems, a method of active differential data synchronization is proposed. We install a service program in source data node in the heterogeneous system to transmit the synchronal data peer to peer, also a differential monitor program is installed in every node of the heterogeneous system to monitor the change of the data and convert the data to an uniform format, then transmit the exchanged data to the wrap page to complete the active synchronal update.

Objective:

Our objective is provide a fundamental resources construction technique which is helpful to the data source integration. We showed how the techniques can be used in the data source integration systems.

1. A taxonomy for querying and integrating heterogeneous and data source there are many traditional and new approaches which can be used for querying and integrating heterogeneous data source . such as federated databases , data warehousing , meta search engines , mediated query systems , etc . one contribution of this work is to characterize theses approaches according to different criteria and then build a taxonomy for them.

2. Single access point for heterogeneous data repositories: the system which implements the requirements we denned for querying and integrating heterogeneous data sources is called Single access point for heterogeneous data repositories . we define the users view on the data, a data model which represents data from the underlying sources in our data , a data model which represents data from the underlying sources in our system and a suitable query language which allows users to formulate a full rang of queries. From fuzzy to exact ones . we define the architecture of such a system and the functionality of each component .

3. Query axplorativeness :in order to process fuzzy queries , a query system needs to "explore" the query and the available data to answer the input query. we introduce a measure called query explorativeness which characterizes the work of the system in order to process such queries and we show its importance .

4. Implementation of query explorativeness in Single access point for heterogeneous data repositories: query axplorativeness is implemented in Implementation of query explorativeness in Single access point for heterogeneous data repositories in a query preprocessing step : using various heuristics , we define algorithm for handling fuzzy queries. Combining database and information retrieval techniques .

2. Heterogeneous Data source integration System

We present the coarse system of single access point for heterogeneous data repositories which follows the usual system see fig 1. Users interact with a user layer in order to submit their queries. The mediation layer has two important into the system. When a query is entered, it is passed to the query transformer component which mainly contains two components the query preprocessor, parses the query and handles the fuzzy part of a query, and the query processor does further processing followed by including the generation of sub queries. One of the main components is the metadata repository which stores information about the underlying data sources. This information is used by system components such as the query transformer, query postprocessor and source registration or by users when formulating queries.

Source Registration requires that the system administrator registers new sources into the system. In order to register a new source and extend the Meta data repository for information characterizing data space information has to be inserted into the metadata repository. Figure 1 illustrates a very simple interface, which helps the administrator to enter the required information. The system can be queried using a simple interface which allows the user to enter SOQL queries on the left side and which gives a simple unified presentation of the results on the right side. Syntactical rules are required to be defined in order to translate SOQL queries into the query language of the source. Additionally, rules for translating the results from the data model of the source into the Single Access point for heterogeneous data Repositories global model is also need

be defined.

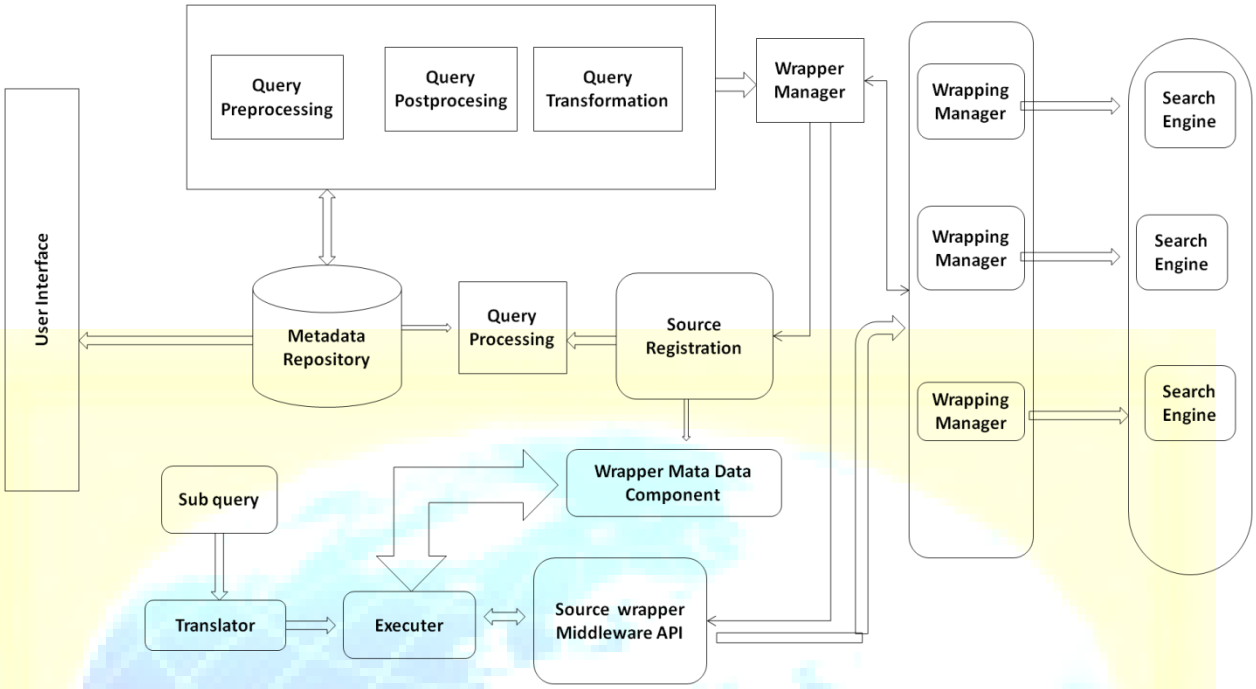

Fig 1: Heterogeneous Data source integration System

Our propose wrapper system provides functionality to view, search and browse the data space, tasks which are fulfilled by the next use cases. SELECT * FROM * WHERE contains ("Database Systems"). The metadata repository, particularly the data space can be browsed in order to formulate more focused queries. For our prototype we have implemented a simple Java applet, represented ill which offers this functionality. Here the data space is represented as a tree with several levels. Our proposed system also supports users or the administrator in order to query the metadata repository. There are two possibilities for this: either the Java applet is used, which also offers a search feature as shown in Figure 1.1 or an SOQL query can be formulated by specifying in the FROM part the keyword

3. Methodology

The prototype has been implemented in the java programming language & the user interface has been partly implemented in Java. Java offers a lot of class libraries and tools, providing rich

built-in functionality. For the development of the prototype, the following class libraries and tools will be used such as Design Patterns. A design pattern in object-oriented systems is systematically names, explains and evaluates an important and recurring design. It helps us to choose design alternatives that make a system reusable and avoid form compromise reusability. [GHJV95] presents some important design patterns for object-oriented design. In particular, we use two of them quite extensively. the visitor and facade design patterns. It helps to reduce the complexity of a system by structuring it into subsystems. Standard Template Library (STL) is a class library that provides a set of well structured generic java components that work together in a seamless way. The library contains five main kinds of components, the most important which we use are algorithm that defines a computational procedure, container that manages a set of memory locations, iterator that provides a means for an algorithm to traverse through a containerLex & Yacc. Lex and Yacc together form a tool which offers a parser generator supporting the automatic generation of parsers written in C or C-\-h. from grammar specifications written in an EBNF (Extended Backus-Naur Form). Lex is the lexical analyzer (or lexer or scanner) that identities the so-called tokens in a structured input. Yacc produces a parser. The prototype contains two main parts: the Single Access point for Distributed heterogeneous data Repositories Server and the wrappers. AbstractTranslator offers functionality for the language translation. For a given language, the administrator specializes it and implements the language translation rules. TraverseQueryTree is a complete implemented class, which does not need to be implemented by the administrator. Abstract Wrapper implementer does not need to know it. AbstractExecutor implements the executor. The results are finally translated into the global model of Single Access point for Distributed heterogeneous data Repositories. The administrator specializes this class for a given source. In Translator Implementation, for each new data source which must be registered into the system, first a new translator class must be defined, which implements the push-down automaton and the translation rules. The implementation consists of a specialization of the class AbstractTranslator and the implementation of the abstract methods which traverse the tree and define the translation rules.

3.1 Executor:

for each new source, the executor must be implemented and which return the results and translates them into the The main classes of the wrapper framework (without attributes and methods) When a new wrapper is implemented, the following steps must be performed

global model (GetNextResultPackage()). Below we grve an example for the method GetNextResultPackageO for a DB2 database.

```
d.lnfoUnit* DB2Executor::GetNextResultPackageO
                    _ currentResu ItPackagePos = 0;
                     if (GetMoreTuplesO) {
                     try {
                              _ currentResultPos++;
                              _ currentkesultl'ackagef'os++;

            return _mapper->GetResult();
```

This method contains two man parts. In the first part, the metadata about the result set is extracted, using the CreateMetaData() method of _m apper.

Wrapper Implementation: Finally, the wrapper for the new data source is a specialization of the class Abstract Wrapper. The constructor of this new class initializes the translator and executor components.

3.2 Data Model:

Since the data model used for The main classes of the wrapper framework (without attributes and methods) When a new wrapper is implemented, the following steps must be performed is ODMG,

extended by some features for semi-structured and unstructured data, we use the java binding which is coarsely. Let us model for example a result set which is returned by querying the

system in the following way: select * from * where name ="Geppert".

We suppose that the system returns two objects, both structured literals: Perslnfo: struct(Personl\lame: string, Address:string) and Personal: struct(Name:string, Vorname:string).

One source will then return a tuple ("Andreas Geppert". "Winterthurerstr. 190, 8057 Zuerich") and the other data source will return a tuple ("Geppert". "A."). The result will be modeled as a d_Set<d_Unit*> which contains two dJJnit's, one of type Perslnfo, one of type Personal, and both types are subclasses of d _Unit

3.3 Extensibility of the Model

The data model needs to be extensible In order to define new types at runtime, when new sources are connected to Single Access point for Distributed heterogeneous data Repositories. Using the modeling primitives defined by the metamodel, we can define new data types. For this, we first give the definition of the meta type d _Type.

class d_Type: public d_Meta_Object

public:

d-Typet);

d _ Type (char * n.char" c=O);

-d_TypeO;

void InsertColiectionType (d _ Ref<d _Collection_Type >&);

void InsertProperty (d_Ref<d]roperty »;

void InsertOperation (d_Ref<d_Operation >&);

void RemoveCollectionType (d_Ref<d_Collection_Type >&);

void RemovePropert) (d _ Ref <d _Property ->&);

void RemoveOperation (d_Ref<d_Operation >&); protected:

d_Rel_Set <d-Collect ion.Type , clnType > _usedlnCollections;

d_Rel_Set <d_Property , clnType > _LlsedlnProperties;

d_Rel_Set <d_Operation, clnType > _LlsedlnOperations;

In short, d_Table_Type contains a list of references to properties and two erations for adding and removing properties. Additionally it contains special erations for tables: union. intersect. product and join. However, one can also define a ational table as a list of d_Property's. If the Single Access point for Distributed heterogeneous data Repositories data model did not offer statical type d_Table_Type, we could define it at runtime in the following way,

d_Type* table = new d_Type (" d_Table_Type");

d_List <d_Ref <d_Property »* tabletype =new d.List <d_Ref<d_Propert)' > >;

d_Ref<d_List <d_Ref<d_Property > > > tablelist = tabletype:

table->lnsertCollectionType ((d_Ref<d-Collect ion.Type »tabletype);

As one can see, ODMG is a model which is powerful enough to support plex data types and to allow the extensibility of the system. However, we had xiend it with several features in order to fulfill our specific requirements.

3.4 Query Language processing :

The class hierarchy for the query language execution is represented As we have defined in the classes which define the query uage are derived from the grammar rules given.. If the query has orm SELECT ... FROM ... WHERE or SELECT ... FROM, then a d_Query object created. class QLVisitor (public virtual void Visitbinary Boolean .J)UCIY nary BooleanQuery") = 0, and the class Ql.Print Visitor is a specialization class of sitar which prints the query on an output stream. In this case, the implementation 'isitBinary_Boolean_Query(...) looks like this:

3.5 Metadata Repository :

There are different kinds of metadata which have to be handled a specific way. First of all, there is information on the type of each data source to be stored. This type information has to be organized in a type tree hierarchy expressing a subtype/super type relationship between data source types. These type nodes have to be accessed efficiently and the set of available nodes has to be edited, i.e. means to add new nodes and remove existing nodes has to be provided. Moreover, information on a data source's schema and its attributes, properties, operations and/or other sub- The data space is represented in the metadata repository using the tools and material metaphor. Finally, the wrappers interfacing the data sources are also considered as a material. Additionally. we have identified the following tools:

- **Type tree tool:** This tool is used for retrieval of type information about the sources. If we consider the data space in the type tree tool manages information about the
- **Data source tool**: This tool is used for management and retrieval of information about data sources, i.e. lower site system in the data space.
- **Path tool:** Since query preprocessing makes extensive use of paths and path operations, we introduced this specialized tool which efficiently manages and retrieves paths in the data space.
- **Wrapper tool**: This tool is used for the management of information about wrappers. For each tool we define a class (TypeTreeTool. DataSourceTool. PathTool and WrapperTool). This class has several subclasses which represent concrete tree nodes:

4. Query Transformation approach and Experimental Results

The design of the query transformation component is of less complexity than the other components. It mainly consists of the of the algorithms. The selection and projection pushing is algorithm by the class Push Visitor. Finally, from the transformed algebra tree, the class As one can remark, the query transformation component handles two trees. the query transformation tree and the algebra tree Since these trees need to be traversed several times in order to implement the processing steps. we make extensive use of the visitor design pattern in this component. The processing of queries the generation of queries for the source fig 1 the processing the combination

of the result for single access point for heterogeneous data repositories we define the following algorithm for computing a final relevance score for a document (used also in the metacrawler search engine. If an underlying source does not return any ranking information for the returned data units we compute a rank in the following way. If a source is in the structured cluster then we consider that each data unit returned has a score of 99 if the source is not in the structured cluster, we index all the data units in the result list and compute a value for each data unit. Consequently, we can assume that each source returns a ranked list and we can then apply the following steps for processing the initial score. map all local scores to [1,999] map largest local score from each source to 999 map other local similarities proportionally. Add local (normalized) scores for data units retrieved from multiple sources. We consider the following example. S1 and S2 are two query systems. S1 return the data unit dl with a score of 99, and S2 with 0.4 then these score are normalized to the [1, 999] interval and dl gets the score 249from S1 and 599 from S2. The finally core for dl is then the addition of the two values 849.

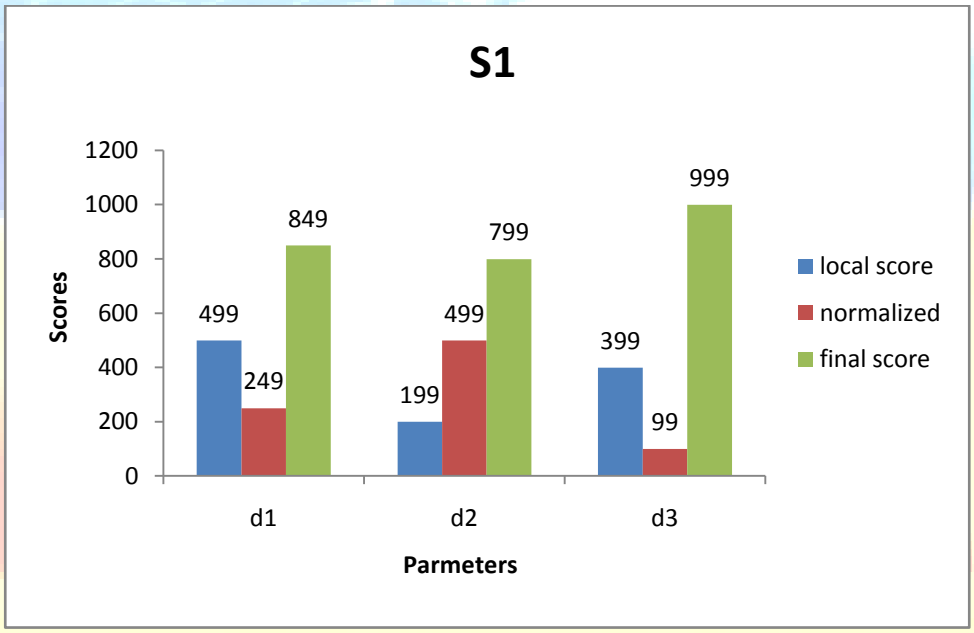

Fig2: Result of Single Access Point approach

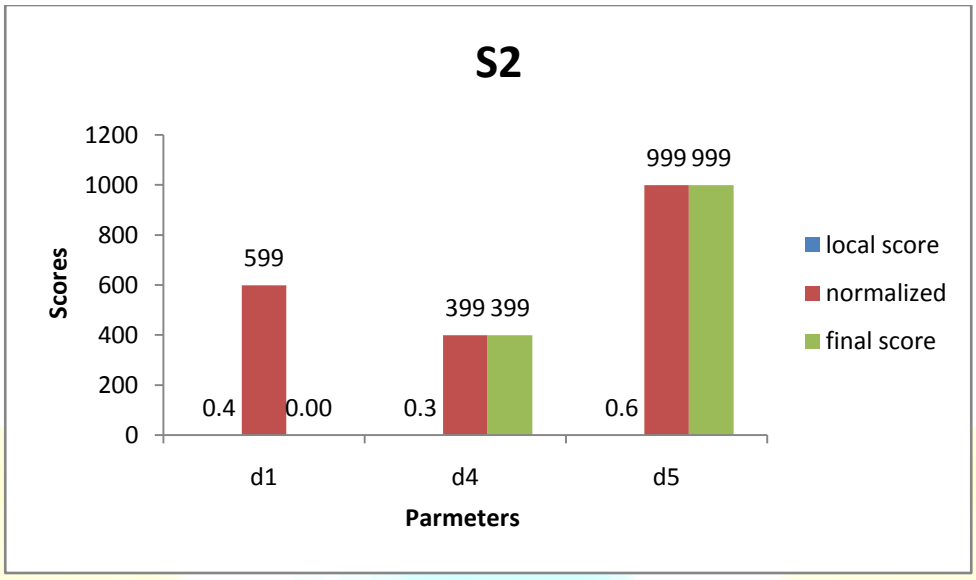

Fig3:Result of Single Access Point approach

5. CONCLUSION

Our research work will mainly focused on study of query structures ,data sources integration, high effectiveness and explorativeness. Query feedback can be used, but they also requires users to have a high knowledge level about the particularities of single access point for heterogeneous data repositories.

Adding a new source at runtime can be easily done but only in case little information about the source needs to be entered into the system. The approach we have presented in this paper defines the functionality of a query service for DBMS. We have considered structurally heterogeneous sources and have shown which query functionality has to be offered by a query service for these kinds of sources. Additionally we have shown how this query functionality can be processed.

REFERENCES

[1] Grazyna Brzykcy, Jerzy Bartoszek, Tadeusz Pankowski. Semantic Data Integration in P2P Environment Using Schema Mappings and Agent Technology. KES-AMSTA'07

[2] M. Akhtar Ali, Alvaro A. A. Fernandes, and Norman W. Paton. MOVIE: An incremental maintenance system for materialized object views. Data & Knowledge Engineering, 47(2):131–166, 2003.

[3] C. Yu and L. Popa, "Semantic Adaptation of Schema Mapping after Schemas Evolve," Proc. Very Large Databases Conf., pp. 1006-1017, 2005.

[4] Gao Xiao-ling, Research and Application on Heterogeneous Data Integration of Mines Based on XML [D], xi'an: Xi'an University of Science and Technology, 2007

[5] M. B. Al-Mourad, W. Alex Gray, and N. Fiddian. Semantically rich materialisation rules for integrating heterogeneous databases. In Proc. of British National Conference on Databases (BNCOD'05), LNCS 3567, pages 60–69, 200

[6] Zhang, Q. Guo, C. S. Iliopoulos, "String Matching with Swaps in a Weighted Sequence", CIS 2004, Springer-Verlag Berlin Heidelberg, LNCS 3314, pp. 698-704, 2004.

[7] Songting Chen, Xin Zhang, and Elke A. Rundensteiner, Member, "IEEE, A Compensation-Based Approach for View Maintenance in Distributed Environments" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 18, NO. 8, AUGUST 2006

[8] Hongzhi Wang Jianzhong Li Zhenying He, "An Effective Wrapper Architecture to Heterogeneous Data Source", Proceedings of the17 th International Conference on Advanced Information Networking and Applications (AINA'03).

[9] Hongzhi Wang, Jianzhong Li, Zhenying He. Compress Communication and Query Process in the Distributed Information Integration System Based on XML. DPCS2002