Journal of Theoretical and Applied Information Technology
10th September 2013. Vol. 55 No.1
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SERVICE ORIENTED ONTOLOGY BASED DATA FEDERATION FOR HETEROGENEOUS DATA SOURCES

HEMA M S[1], CHANDRAMATHI S[2]
[1]*Department of Computer Science and Engineering, Kumaraguru College of Technology*
[2]*Department of Computer Science and Engineering, Hindustan College of Engineering and Technology*
INDIA
Email: [1]ghema_shri@yahoo.co.in, [2]chandrasrajan@gmail.com

ABSTRACT
Data integration is inevitable in current distributed heterogeneous database environment. The existing data integration system uses data federation to integrate the distributed data without duplicating the data sources or creating a new integrated data store for analysis and decision support. The contemporary data federation techniques convert the data sources to XML dataset and integrate the data. These systems do not resolve all conflicts efficiently. This paper proposes a layered Service Oriented Ontology based Data federation Architecture [SOOD] and hybrid ontology based data federation methodology over the SOOD architecture. This converts the metadata of data sources into OWL description for constructing global ontology using Local as View [LAV] approach for data federation. In the proposed method global ontology construction process uses the OWL description to resolve all schema level heterogeneities of data sources and guarantees quality data federation. The heterogeneities are classified in to two (schematic and data) and are resolved at two different stages of proposed ontology based data federation methodology. The storage infrastructure of SOOD architecture aids to store and fetch the results swiftly. Experimental results shows that the proposed methodology is relatively efficient than the conventional hybrid ontology based data federation methodology in recall and other quality parameters like response time and reusability. It is observed that the recall of SOOD system is increased by 10% and response time is reduced by 20% than the conventional hybrid ontology system.

Keywords: *Data Federation, Data Integration, Ontology, Resolving Heterogeneity, Service Oriented Computing.*

1. INTRODUCTION

In current digital era, huge amount of data are stored in storages that are geographically distributed. To analyze and extract information from these distributed data sources for decision making, the sources are to be integrated. Data integration facilitates query over autonomous and heterogeneous data sources through a common and uniform schema. The data integration techniques like data consolidation, data propagation and data federation are used for collating data from various sources.

Data federation is a process of creating an integrated virtual table (view) from multiple heterogeneous data sources without any replication of data sources. The integrated table is used for data analytic applications. Enterprise Information Integration [EII] technology supports data federation. To construct integrated view the data federation needs to resolve a) Syntactical heterogeneity b) Data model heterogeneity and c) logical heterogeneity. The data federation are classified into i) Loosely Coupled federation, 2) Federated Database and 3) Mediator based federation. A simple data federation model is shown in Figure 1.
The existing data federation systems convert the native data sources into XML files to resolve heterogeneities [3][4][7]. These XML file are less expressive, use reverse functional properties and result in relatively poor integrated data.

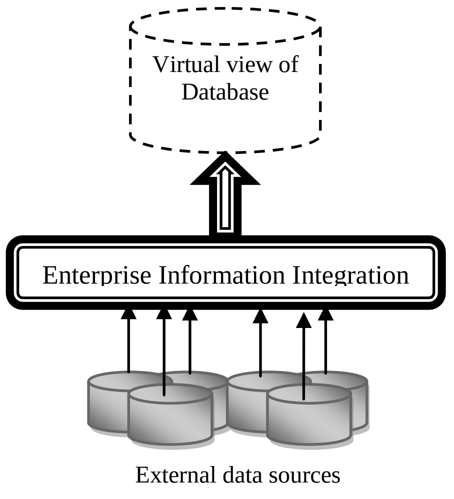


Figure-1 Data Federation

Ontology is generally regarded as a designed artifact consisting of a specific shared vocabulary used to describe entities in some domain of interest, as well as a set of assumptions about the intended meaning of the terms in the vocabulary. The concept of ontology was originated from artificial intelligence literature on *Declarative Knowledge*, which is concerned with the formal symbolic representation of knowledge. Ontology based Data Integration involves the use of ontology(s) to effectively combine data and/or information from multiple heterogeneous sources. The effectiveness of ontology based data integration is closely tied to the consistency and expressivity of the ontology used in the integration process. The ontology based data integration is classified into three. They are Single ontology, Multiple ontology and Hybrid ontology [3][6].

This paper proposes a Global Local as View [GLAV] ontology based data federation over SOOD [Service Oriented Ontology based Data Federation] architecture which extends a typical data federation architecture. This architecture contains User Interface Layer, Knowledgebase Layer, Federation Layer and Data Source Layer. This architecture enables the proposed system to resolve heterogeneities of data sources in its native form instead of XML wrapper. The system obtains schema of different data sources and converts them into OWL description are called local ontologies. These local ontologies are used to build global ontology resolving schema level conflicts like Naming conflict, Semantic conflict and Schema isomorphism conflict.

The system receives user request through interface layer and searches the cached result repository of Knowledgebase layer. If the cached result repository contains results for the input, then the results are communicated to the user; else the query is send to Federation layer for query processing. The query processing service decomposed and converts the query into native database acceptable form and the converted queries are executed in respective data storage servers. The results generated from different storage servers are populated in the virtual table using result integration service. The integrated data is communicated to user and converted into XML data and stored in cache repository for future reference.

The proposed ontology based data federation methodology over SOOD architecture resolve schema level and data level conflicts and integrated data relatively better than the existing data federation methodologies. Experimentations were carried out over different datasets and the results are analyzed. Rest of this paper is organized as follows, Section 2 discusses the related literature review of SOOD architecture and the proposed methodology are detailed in Section 3, Section 4 provides implementation details and experimental results and concluding remarks of proposed system are given in Section 5.

2. RELATED RESEARCH

Gongzhu Hu has created a global view from local view by resolving a subset of structural and semantic discrepancies of local database. The native data sources are not converted to XML for integration. In this methodology the global schema is constructed as union of local schema of respective data sources (Gongzhu Hu, 2006) [1]. Yumeng Zhao et al proposed Conflict Resolution Model to resolve data level and semantic level using mapping formulas and conflict information among heterogeneous data. The model also employs handlers to correct queries (Yumeng Zhao, Shidong Zhang, Zhongmin Yan, 2003)[2]. Laomo Zhang et al extended hybrid ontology approach to integrate data by representing global ontology, local ontology and mapping rules in ontology web language. The three stages of this method are building the shared vocabulary, building local ontologies and defining mapping (Laomo Zhang, Ying Ma, Guodong Wang, 2009)[3].

Ze Hua and JianMin Ban have expressed ontology in RDF schema and constructed using global-as-view approach by merging individual local ontologies , which represent XML source schemas (Ze Hua, JianMin Ban, 2010)[4]. Joao C. Pinheiro

et al have proposed a three-level ontology-based architecture for data integration. This architecture takes a SPARQL query submitted over domain ontology and rewrites it into sub-queries over multiple data sources. The query's result is obtained by the proper combination of data obtained from these sub queries (Joao C. Pinheiro et al, 2010)[5].
Ontology based approach for semantic integration was proposed by Michel Gagnon which integrates heterogeneous data sources with local to global ontology mapping. This method provides interoperability between different heterogeneous data sources (Michel Gagnon, 2007)[6]. Jinpeng Wang proposed methodology to bridge the gap between SPARQL to SQL using graph pattern transformation algorithm (Jinpeng Wang et al, 2010)[14].
The existing methods do not ensure efficient result integration, automated mapping table construction, integration of data from its native form [3][4][7]. The RDF ontology methods are less expressive, all attributes of local schema can't be mapped with GAV and employs reverse functional properties for data integration. And the existing system doesn't have feature to update the integrated results in knowledgebase layer if the distributed data sources are modified or updated. To address the above issues the conventional data federation architecture is extended by adding unique repositories to store the results for improving response time, a service to update the integrated data store and the schema and data level heterogeneities are resolved in two stages of data federation which are detailed in the proposed system.

3. PROPOSED SYSTEM

The proposed SOOD [Service Oriented-Ontology based Data federation] architecture for data integration as shown in Figure 2. This architecture consists of four layers a) User Interface Layer, b) Knowledgebase Layer, c) Federation Layer and d) Data Source Layer. Each layer has mutually exclusive task to perform.
The sequence of execution for various processes of proposed architecture for data federation is shown in Figure 3. The user sends the query through the user interface for federation. The knowledgebase layer verify the cache, if query is available the corresponding result is communicated to user through user interface else the query is stored in cache and it is passed to data federation layer.

*Figure – 2 Architecture Of SOOD**

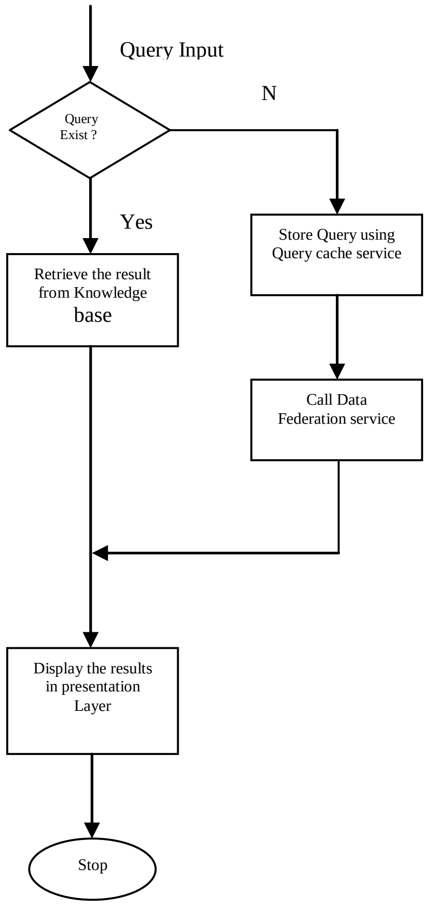

Query Input
Query Exist ?
N
Yes
Retrieve the result from Knowledge base
Store Query using Query cache service
Call Data Federation service
Display the results in presentation Layer
Stop

Figure – 3 Sequence Of Execution Of SOOD Architecture

3.1. User Interface Layer

The user interface layer collects input from the user and displays the result to the user in user understandable form. This layer contains two modules i) Input Module accepts query from user and validate it for processing and ii) Result display Module displays the integrated results to user for the requested query.

3.2. Knowledgebase Layer

The knowledgebase layer preserves the results of executed query for future reference in a repository. The input query is verified with the entries of knowledgebase repository. The result of matching query is extracted and displayed to the user. The other queries are carried to federation layer for processing.

Journal of Theoretical and Applied Information Technology
10th September 2013. Vol. 55 No.1
© 2005 - 2013 JATIT & LLS. All rights reserved.
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Knowledgebase layer contains i) Query Cache Module and ii) Result cache module to store query and corresponding result in repository. For instance a request *"Extract courses of all departments whose pass percentage is greater than 75%"* is provided by the user. The request is stored in knowledgebase repository as shown in Table 1. The knowledgebase repository uses B+ tree to store and search. This layer contains Trigger module which removes the queries and respective results from the repository during any updates in related data sources.

Table – 1 Knowledgebase Repository

Query ID	User Request	Result
U10032	*Extract courses of all departments whose pass percentage is greater than 75%*	<Course>***CS1292***<\course><res>***82***<\res> <Course>***CE1286***<\course><res>***78***<\res> <Course>***ME1357***<\course><res>***94***<\res>

3.3. Federation Layer

Federation layer is segregated in three sub layers i) Ontology layer for mapping global and local ontology, ii) Query Processing layer for query decomposition, query execution and query optimization and iii) Result integration and heterogeneity resolving layer for integrating results and to remove data level heterogeneities of data during integration.

Ontology layer consists of three components a) Application ontology service, b) Domain ontology service and c) Mapping services as shown in Figure 2. Application ontology and Domain ontology is called local ontology and global ontology respectively.

The local ontology is obtained from local schema using web ontology language [OWL]. An OWL description for course schema is shown in Table 2. This defines OWL classes, relationship, properties and constraints.

Global ontology is constructed using Local as View [LAV]. The LAV combines the concepts and attributes with same semantics and adds the remaining to create global ontology using Protégé. The global ontology resolves synonyms, schematic discrepancies and schema isomorphism. Mapping service contains rules or guidelines to map local and global ontology.

Table – 2 OWL Description For Course Schema

Local ontology-1: OWL description
*<owl:Class rdf:about="**#courseid**">* *<rdfs:subClassOf rdf:resource="**#course**"/>* *</owl:Class>* *<owl:Class rdf:about="**#coursename**">* *<rdfs:subClassOf rdf:resource="**#course**"/>* *</owl:Class>* *<owl:Class rdf:about="**#numberofcredits**">* *<rdfs:subClassOf rdf:resource="**#course**"/>* *</owl:Class>* *<owl:Class rdf:about="**#course**"/>*

Three steps involved in global ontology construction are analyzing local ontologies, creating global ontology and creating mapping table for attributes of global and local ontologies. Ontology mapping of the proposed methodology over SOOD architecture is shown in Figure 4.

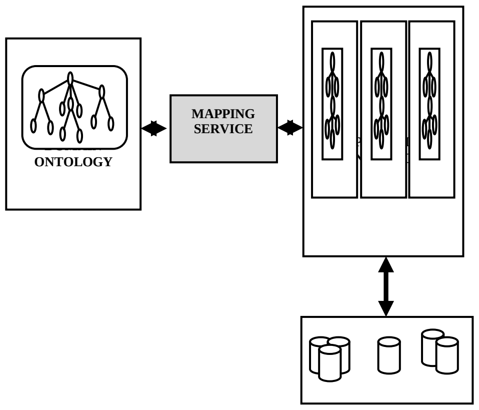


Figure – 4 Ontology Mapping

Analyzing local ontology involves extraction of relationship between entities, semantics of attributes and domain of attributes. Using extracted information global ontology is created based on guidelines shown in Table 3.

*Table – 3 Mapping Guidelines Table**

Query processing layer contains Query decomposing module and Query execution module. The query is decomposed using mapping rules in the mapping service. The decomposed sub queries are sent to respective data source for execution by query execution service. The results of sub queries are passed to heterogeneity resolving module to resolve i) Date representation, ii) Data units and iii)

Data precision. The resolved results are integrated by result integration module as shown in Equation 1.

$$Ir = rdb1 \; U \; rdb2 \; U \ldots\ldots U \; rdbi \ldots. U_{rdbn} \quad (1)$$

Where Ir: Integrated results, rdbi: results of i^{th} data source

The heterogeneity resolving and result integration layer resolve data level conflicts like data scaling and data precision during result integration and schema level conflicts like semantic, structural discrepancies and name conflicts are resolved during ontology mapping shown in Table 4.

*Table – 4 Resolving Heterogeneity**

4. IMPLEMENTATION AND EXPERIMENTAL RESULTS

To experiment the proposed system the following web services are implemented; Query cache service, Result cache service, Triggering service, Query decomposition service, Query execution service, Application ontology Service, Global ontology Service, Mapping service, Heterogeneity resolving service and Result integration service. Experimental dataset contains five heterogeneous data sources belong to university domain. Initially the data sources are populated with 200 records and gradually increased till each data source contains 1400 records. Subset of data sources and relations are shown in Figure 5.

Figure – 5 Data Sources And Relations

4.1. Performance Measure

Recall is a measure of information system used to appraise the relevance of extracted data. The Recall is calculated using the equation shown in Equation 2.

$$Recall = \frac{((No.of\ relevant\ records) \cap (No.\ of\ records\ retrieved))}{(Total\ No.of\ records\ )} \quad (2)$$

Recall is measured by passing different types of queries to the data sources. The average recall measure for proposed system and conventional hybrid ontology system with various types of queries are presented in Table 5. The schema and data conflict resolving process of proposed SOOD system improves the recall rate and is relatively better than the conventional hybrid ontology system. In conventional hybrid ontology method the queries are mapped to GAV which would result in data loss in integrated data store due to missing attributes of data sources [1][4][5]. The recall rate of SOOD methodology will increase if data qualities of respective data sources are good and complete. The response time for proposed SOOD system with conventional hybrid ontology system is show in Figure 6. Response time of proposed system with and without cache results is shown in Figure 7.

Table – 5 Recall Rates Of Proposed Ontology Based Data Federation System And Conventional Hybrid System For Different Types Of Queries

Query Type	Average Recall Rate for proposed ontology based data federation (in %)	Average Recall Rate for conventional hybrid ontology system (in %)
Queries with 'where' clause	92%	67%
Queries with relational operators(>,<,<=,>=)	89%	79%
Queries with logical operators	90%	78%

Experiment is conducted to evaluate the effective response time of proposed system for various size of data sources. Effective response time for results retrieved from knowledgebase layer is calculated using Equation -3. The cache hit rate is found using Equation 4.

*Figure – 6 Response Time For Proposed System And Conventional Hybrid Ontology Based System**

*Figure – 7 Response Time For Proposed SOOD System With And Without Cache Repository**

$$Effective\ Response\ time = (cache\ hit\ rate * response\ time) \quad (3)$$

$$cache\ hit\ rate = \frac{No.of\ results\ retrieved\ from\ knowledgebase\ layer}{Total\ number\ of\ queries} \quad (4)$$

Similarly effective response time for results that are retrieved from federation layer is calculated using Equation – 5. The cache miss rate is found using

Equation 6. The effective response time for query set is shown in Table - 6.

$$Effective\ Response\ time = (cache\ miss\ rate * response\ time) \quad (5)$$

$$cache\ miss\ rate = \frac{No. of\ results\ not\ retrieved\ from\ knowledgebase\ layer}{Total\ number\ of\ queries} \quad (6)$$

*Table – 6 Effective Response Time**

The effective response time of SOOD methodology will improve if overall cache hit rate increases. The triggering service of SOOD ensures that the knowledgebase contains updated results.
Response time of SOOD system and conventional hybrid ontology method for 10 different query set is displayed in Table 7 and their average performance is shown in Figure 8. The performance of SOOD methodology is relatively better than the conventional hybrid ontology method [1][3][5][6][7].

*Table – 7a Response time for 10 different query set in conventional hybrid ontology method**

*Table – 7b Response time for 10 different query set**

*Figure – 8 Average Response time**

5. CONCLUSION

The knowledgebase layer of the SOOD architecture caches the integrated results of user request for future reference and to improve response time. The proposed ontology based data federation integrates heterogeneous data sources resolving schema and data level heterogeneities over the SOOD architecture. The experimental result shows that the proposed SOOD method consumes 20% less response time than that of the conventional hybrid ontology method to retrieve data. The recall rate of proposed system is 10% higher than that of conventional hybrid ontology approach.

REFERENCES

[1] Gongzhu Hu (2006), ‘Global Schema as an inversed view of Local Schemas for Integration’, *International Conference on Software Engineering Research*, SERA’06.

[2] Yumeng Zhan, Shidong Zhang, Zhongmin Yan (2009), ‘Ontology – based Model for Resolving the Data-level and Semantic-level Conflict’, *Internationa Conference on Information and Automation.*

[3] Laomo Zhang, Ying Ma, Guodong Wang (2009), ‘An Extended Hybrid Ontology Approach to Data Integration’, *International Conference on Biomedical Engineering and Informatics*, BMEI ’09, pp 1-4..

[4] Ze Hua, JianMin Ban (2010), ‘Ontology-based Integration and Interoperation of XML Data’, *Sixth International Conference on Semantics*, Knowledge and Grids.

[5] Joao C Pinheiro, Vania M. P. Vidal, Jose A. F. Macedo, Eveline R. Sacramento, Marco A. Casanova, Fabio A. M. Porto (2010), ‘Query Processinf in a Three-Level Ontology Based Data Integration System’, *International Conference of Information Integration and Web based Application Service*, iiWAS2010, pp 283-290.

[6] Michel Gagnon (2007), ‘Ontology-Based Integration of Data Sources’, *International Conference on Information Fusion*, pp 1-8.

[7] Li Yan-heng, Zhang Jin, Peng Ying (2010), ‘A study on XML and Ontology-based Web Data Integration’, *International Conference on Computer and Information Technology*, CIT 2010.

[8] Andreas Langegger (2008), ‘Virtual Data Integration on the Web-Novel Methods for Accessing Heterogeneous and Distributed Data with Rich Semantics’, *International Conference on Information Integration and Web based Integration System*, iiWAS2008, pp 559-562.

[9] Dejing Dou, Paea LePendu, Shiwoong Kim, Peishen Qi (2006), Integrating Databases into the Semantic Web through and Ontology – based Framework’, *International Conference on Data Engineering Workshops*, ICDEW’06.

[10] Stuart E. Madnick, Richard Y. Wang, Yang W. Lee, Hongwei Zhu (2009), ‘Overview and Framework for Data and Information Quality Research’, *ACM Journal of Data and Information Quality*, Vol. 1, No. 1, Article 2.

[11] L. Bradji, M. Boufaida (2011), ‘Users Expectations Feedback Consistency as a first step for a better Data Quality’, *Journal of Theoretical and Applied Information Technology*, Vol. 33, No. 1, pp 58-68.

[12] R. Kavitha Kumar, RM. Chandrasekaran (2011), 'Attribute Correction-Data Cleaning using Association rule and Clustering Methods', *International Journal of Data Mining & Knowledge Management Process*, Vol. 1, No. 2, pp 22-32.

[13] Richard Y. Wang, Veda C. Storey, Christopher P. Firth (1995), 'A Framework for Analysis of Data Quality Research', *IEEE Transaction on Knowledge and Data Engineering*, Vol. 7, No. 4, pp 623-639.

[14] Jinpeng Wang, Yafie Zhang, Zhuang Miao, Jianjiang Lu (2010), 'Query Transformation in Ontology-based relational Data Integration', *Asia-Pacific Conference on Wearable Computing Systems*, APWCS 2010.

[15] Rodrigo Fernandes Calhau, Ricardo de Almedia Falbo (2010), 'An Ontology-based Aproach for Semantic Integration', IEEE International Enterprise Distributed Object Computing Conference, EDOC 2010.

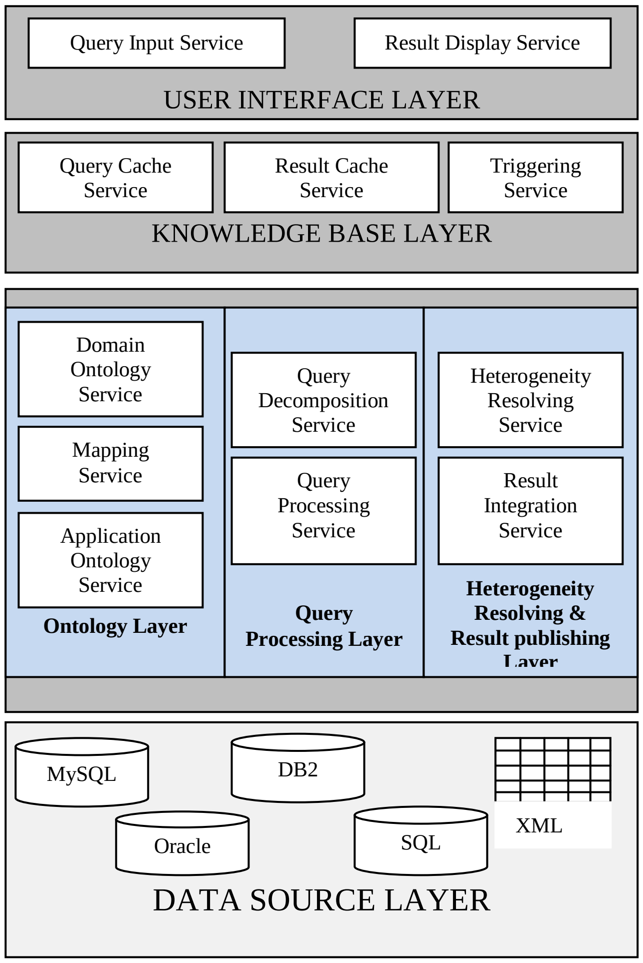


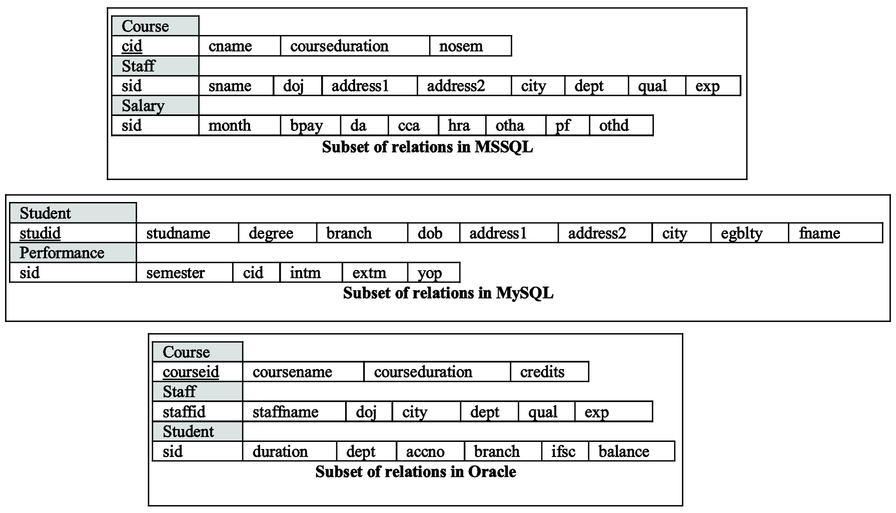
Figure – 2 Architecture of SOOD

Table – 3 Mapping Guidelines table

Let R1 and R2 be any related relations and attributes a1, a2, a3 ϵ R1 and b1, b2, b3 ϵ R2 in local ontology. The global ontology G is created with attributes g1, g2, g3, …	
Local ontology Constraints	**Domain ontology Mapping Guidelines**
Same semantics *If a1 ≡ b1 and a2 ≡ b2 and a3 ≡ b3*	$G \longleftarrow R1 \cup G \longleftarrow R2$
Structural discrepancies *If a1 . a2 ≡ b2 or concat(a1, a2) ≡ b2* *If a1 ≡ b1 . b2 or a1 ≡ concat(b1, b2)*	$G(g2, g3) \longleftarrow R1(concat(a1, a2), a3) \cup G(g2, g3) \longleftarrow R2(b2, b3)$ $G(g1, g2) \longleftarrow R1(a1, a2) \cup G(g1, g2) \longleftarrow R2(concat(b1, b2), b3)$
Schema isomorphism *If a1 ≡ b2 and a2 ≡ b3 and a3 ≡ b1*	$G \longleftarrow R1(a1, a2, a3) \cup G \longleftarrow R2(b2, b3, b2)$
Independent relations *If a1 ≡ b1 and a2 ≡ b2 and a3 ≠ b3*	$G \longleftarrow R1(a1, a2) \cup G \longleftarrow R2(b1, b2) \bowtie R1(a3) \bowtie R2(b3)$

Table – 4 Resolving Heterogeneity

Heterogeneity type	Local Schemas	Resolving Heterogeneity	Stage
Name conflict	L1:Course(courseid char, coursename varchar, numberofcredits int) L2:course(cid char, coursename varchar, numberofcredits int)	L1:courseid and L2:cid are semantically equivalent and are mapped to G:CourseID	Ontology Mapping
Semantic conflict	L1:Student(rollno char, stuname varchar, courseid char) L2:Studentrel(rollno varchar, stuname varchar, courseid char) L3:student(rollno int, stuname varchar, courseid char)	L1:rollno, L2:rollno and L3:rollno are type casted to varchar and mapped to G:RollNo	
Structural discrepancy	L1:performance(monthandyear date, rollno varchar, courseid varchar, grade int) L2: :performance(monthandyear date, rollno varchar, course1 varchar, grade1 int, course2 varchar, grade2 int, course3 varchar, grade3 int)	L1:courseid and L2:course1, course2, course3 are mapped to G:CourseID	
Schema isomorphism	L1:staff(staffed varchar, staffname varchar, address varchar, email varchar, mobileno varchar) L2: staff(staffed varchar, staffname varchar, address1 varchar, address2 varchar, address2 varchar, town varchar, city varchar, email varchar, mobileno varchar)	L1: address is mapped to G:Address and L2:address1, address2, address3 are concatenated to map with G:Address	
Data Scaling	L1:fees(rollno varchar, feesindollars decimal(7:2), year date) L2: fees(rollno varchar, feesinrupees decimal(7:2), year date) L3: fees(rollno varchar, feesineuros decimal(7:2), year date)	L1:feesindollars, L2:feesinrupees and L3:feesineuros are converted to same units and mapped to G:FeesAmount	Result integration
Data Precision	L1:performance(monthandyear date, rollno varchar, courseid varchar, grade int) L2:performance(monthandyear date, rollno varchar, courseid varchar, grade char)	L1:grade and L2:grade are converted to same precision and mapped to G:Grade	
L1, L2 and L3- Local ontologies G – Global ontology			



Course									
cid	cname	courseduration	nosem						
Staff									
sid	sname	doj	address1	address2	city	dept	qual	exp	
Salary									
sid	month	bpay	da	cca	hra	otha	pf	othd	

Subset of relations in MSSQL

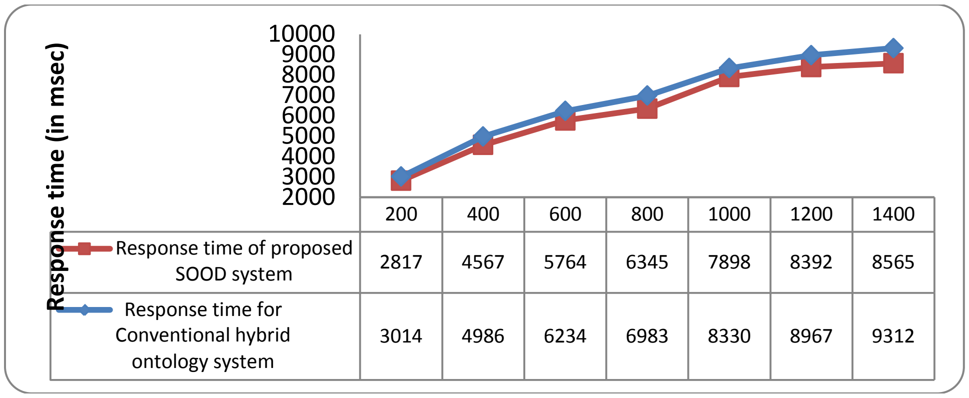
Student									
studid	studname	degree	branch	dob	address1	address2	city	egblty	fname
Performance									
sid	semester	cid	intm	extm	yop				

Subset of relations in MySQL

Course							
courseid	coursename	courseduration	credits				
Staff							
staffid	staffname	doj	city	dept	qual	exp	
Student							
sid	duration	dept	accno	branch	ifsc	balance	

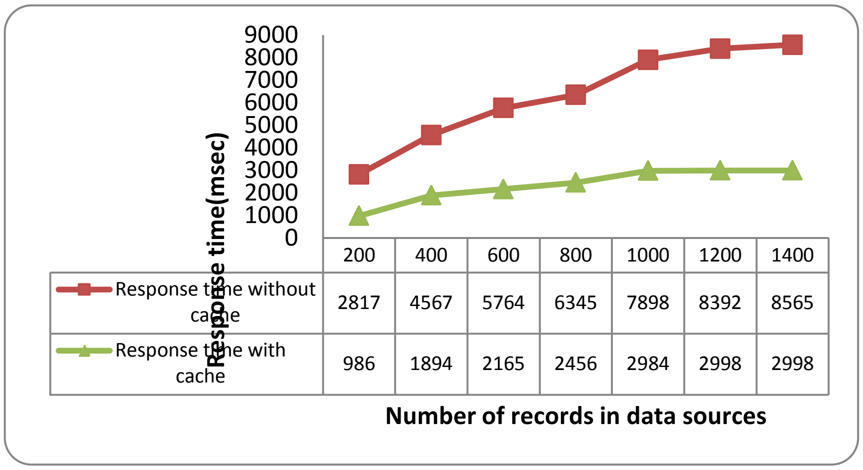
Subset of relations in Oracle

Figure – 5 Data sources and relations



	200	400	600	800	1000	1200	1400
🟥 Response time of proposed SOOD system	2817	4567	5764	6345	7898	8392	8565
🔷 Response time for Conventional hybrid ontology system	3014	4986	6234	6983	8330	8967	9312

Figure – 6 Response time for proposed system and conventional hybrid ontology based system



	200	400	600	800	1000	1200	1400
🟥 Response time without cache	2817	4567	5764	6345	7898	8392	8565
▲ Response time with cache	986	1894	2165	2456	2984	2998	2998

Figure – 7 Response time for proposed SOOD system with and without cache repository

Table – 6 Effective Response time

Query set	Response Time (msec)	Effective Response time (msec)	
		Hit Rate 0.3	**Miss Rate 0.7**
Q1	2885	865.5	-
Q2	8267	-	5786.9
Q3	8346	-	5842.2
Q4	8286	-	5800.2
Q5	8346	-	5842.2
Q6	2881	864.3	-
Q7	8753	-	6127.1
Q8	8121	-	5684.7
Q9	2788	836.4	-
Q10	8334	-	5833.8

Journal of Theoretical and Applied Information Technology
10th September 2013. Vol. 55 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table – 7a Response time for 10 different query set in conventional hybrid ontology method

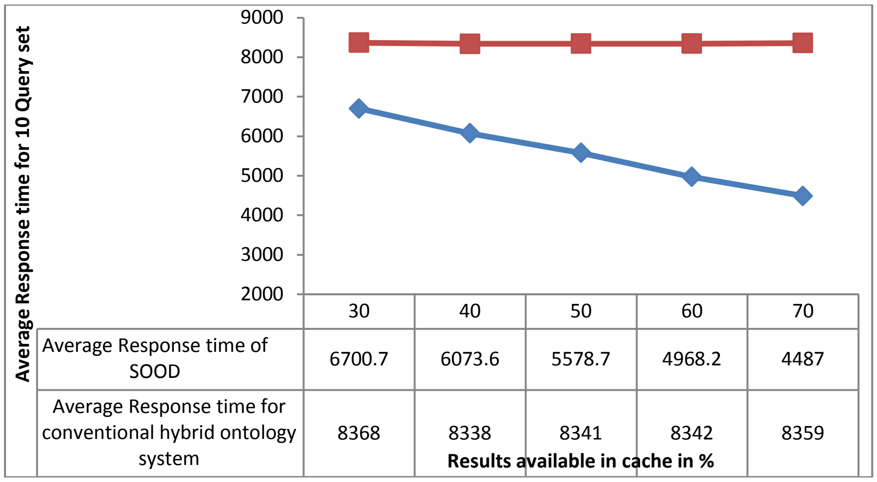
Results available in cache (%)	Response Time (msec)											
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total	conventional hybrid ontology system*
30	8357	8345	8346	8432	8346	8320	8346	8451	8399	8334	83676	8367.6
40	8389	8325	8239	8430	8345	8324	8432	8299	8239	8361	83383	8338.3
50	8329	8291	8339	8421	8245	8299	8411	8312	8341	8421	83409	8340.9
60	8356	8233	8311	8322	8323	8345	8423	8389	8367	8352	83421	8342.1
70	8329	8329	8311	8341	8328	8410	8332	8451	8345	8411	83587	8358.7

Q1-Q10 – query set submitted for data federation.
* Average response time for 10 query set

Table – 7b Response time for 10 different query set

Results available in cache (%)	Response Time (msec)													% of improvement
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total	Avg. SOOD	conventional hybrid ontology system*	
30	2885	8267	8346	8286	8346	2881	8753	8121	2788	8334	61135	6700.7	8368	19.9
40	2823	8325	8239	8231	8232	2799	2821	8223	2682	8361	60736	6073.6	8338	27.1
50	2824	2911	8339	8284	8229	2875	2881	8212	2811	8421	55787	5578.7	8341	33.1
60	2782	2882	8214	8199	7992	2731	2839	2819	2872	8352	49682	4968.2	8342	40.4
70	2867	2891	2821	8341	8324	2821	2821	2821	2837	8326	44870	4487	8359	46.3

Q1-Q10 – query set submitted for data federation.
* Average response time for 10 query set



	30	40	50	60	70
Average Response time of SOOD	6700.7	6073.6	5578.7	4968.2	4487
Average Response time for conventional hybrid ontology system	8368	8338	8341	8342	8359

Figure – 8 Average Response time