ROHDIP: Resource Oriented Heterogeneous Data Integration Platform

Wael Shehab
Computers & Control Dept.
Faculty of Engineering Tanta Univ.
Tanta, Egypt

Sherin M. ElGokhy
Computers & Control Dept.
Faculty of Engineering Tanta Univ.
Tanta, Egypt

ElSayed Sallam
Computers & Control Dept.
Faculty of Engineering, Tanta Univ.
Tanta, Egypt

***Abstract*—During the last few years, the revolution of social networks such as Facebook, Twitter, and Instagram led to a daily increasing of data that are heterogeneous in their sources, data models, and platforms. Heterogeneous data sources have many forms such as the www, deep web, relational databases systems, No-SQL database systems, hierarchal data systems, semi-structured files, in which data are usually allocated on different machines (distributed) and have different data models (heterogeneous).**

Large-scale data integration efforts demonstrate that their most valuable contribution is implementing a data integration platform that provides a uniform access to the heterogeneous data sources, as well as the different versions of data reported by the same data source over time. Furthermore, the platform must be able to integrate data from a broad range of data authoring devices and database management systems. It also should be accessible by almost types of data querying devices to ensure globally querying the integration platform from any place on earth anytime and receiving the query result in any data format.

In this paper, we create a resource oriented heterogeneous data integration platform (ROHDIP) that facilitates the data integration process and implements the objectives discussed above. We use the resource oriented architecture ROA to support the uniform access by most types of data querying devices from anywhere and to improve the query response time.

Keywords—Data Integration; Data heterogeneity; SOA; ROA; Restful; ROHDIP

I. Introduction

The enduring utilization of information technology raises data sharing as a challenging problem for many enterprises. Most enterprise information management systems adopted the opinion of establishing isolated database management systems in departments that may be geographically dispersed or have different business type to improve production, management, and efficiency. However, these systems are developed using different software companies at various times, on different platforms as well, which inevitably will lead to the coexistence of heterogeneous databases [1]. Heterogeneous data are often collected from an unknown or an unlimited number of sources in different formats. Two types of data heterogeneity; namely, structural heterogeneity and semantic heterogeneity are counted. In structural heterogeneity, the information systems store data in several structures. While semantic heterogeneity concerns with both the data item content and its intended meaning. The rapidly increasing number of structured, semi-structured data sources results in a crucial need for uniform and flexible query interfaces to access data that are distributed on heterogeneous and autonomous sources [2] [3] [4] [5].

Data integration system allows users to specify what information is needed without providing detailed instructions of the methodology followed to obtain that information or even specifying its location. In order to have the capacity to do so, data integration system must be able to do the following process: communication and interaction with data sources, unifying different queries in requester specifying vocabulary (ontology) across multiple autonomous, distributed and heterogeneous data sources, mapping techniques between requester ontology and the data source ontology, extracting information from the query with respect to the target data sources, and finally translating the query results to the requester vocabulary [4].

Recently, many approaches to data integration were developed including manual integration, application-based integration, middleware data integration, physical data integration and virtual integration. In manual integration (common user interface), users manage all relevant information, accessing all the source systems and there is no unified view exists for the data. Application-based integration requires the particular applications to achieve all the integration efforts; therefore, this approach is manageable only in case of a limited number of applications. The approach that transfers the integration logic from particular applications to a new middleware layer is called middleware data integration. However, this approach does not ensure achieving the practical requirements. Physical data integration usually creates a new system that copies the data from different source systems to be stored and managed independently of the original system. The most well-known implementation of this approach is called data warehouse (DW) [6][7] which combines data from different sources (such as mainframes, databases, flat files). However, the need for a separate system to handle the vast volumes of data constitutes demerit of this approach. The final approach is virtual integration which leaves data in the source systems and defines a set of views to provide the customer with a unified view of the whole enterprise. For example, when we need to query specific data, it will be retrieved only from its data source [8] [9].

Virtual integration approach has several advantages that make it one of the most successful data integration approaches. It succeeds to propagate the data update from the source system to the integration system with almost zero latency.

Also, virtual integration has no need to copy any data from the data source to the integration system. Also, it does not need to unify the distributed data sources. Based on that, this paper is concerned with proposing an algorithm that adopts the virtual integration approach.

The rest of this paper is organized as follows: the related work is discussed in section II. Section III illustrates the problem statement and presents the proposed framework. The experimental results are exhibited and analyzed in section IV. Finally, Section V concludes the paper ideas and suggests future research ideas.

II. Related Work

Several data integration platforms have been developed to provide a uniform query interface that has the capability to query the heterogeneous data sources [10] [11].

Specifically, an extensive wide variety of methods is proposed to achieve virtual integration approach, each of which was targeting the same goal but with its autonomous way. These methods are categorized into two approaches: Global-As-View (GAV) and Local-As-View (LAV). GAV produces a top abstract level that constitutes a single mediated schema described as views (mappings) of all local data sources [12] [13], while LAV describes the local data sources as views over a global schema [2] [14]. After that, many approaches have been derived from LAV and GAV [11] [15] making the best use of these two modern technologies, such as SOA “Service Oriented Architecture” that is used in implementing dynamic and flexible integration systems; namely, Service Oriented Data Integration systems. Then, various data integration frameworks based on SOA have been developed in the last few years such as SODIA architecture [11]. SODIA merges the data at various, distributed, heterogeneous and autonomous data sources into a single dynamic view. Service providers publish their data sources as data access services, which may be detected instantly at the time they are needed and released after use. Hence, variations of organization structures, backend data sources, data structures, or semantics could be managed and potentially the maintenance cost is reduced.

In 2009, an architecture for "Internet of Things" has been proposed to connect millions of different devices together based on service-oriented approach [16]. The architecture hides the heterogeneity of hardware, software, data formats and communication protocols. The specifications of the architecture support open and standardized communication via web services at all layers. Services abstract all functionality offered by networked devices. A runtime for the execution of the composed services was provided.

After that, Sanz et al. proposed an approach to integrate several technologies, such as the JSF, Spring and Hibernate frameworks in a multilayer architecture. SOA architecture provides services to allow collaborative work, using the independent development of components in different layers. The approach relies on developing a global software system where the presentation layers for different end devices are separated from the business logic layer, whose services are reused for three types of user interfaces without changing the code [17].

The challenges of interconnection and communication of different protocols between heterogeneous systems have been investigated in 2010 [18]. An integration platform is constructed to achieve the synchronization and transformation of data between heterogeneous systems through registering, mapping the various service components and constructing the SOA framework of enterprise based on the service component. The platform uses XML as a middleware for mapping several data sources into a unified model depending on a set of mapping transformations. The element of each mapping model has one service component which indicates the source or destination of elements. So, a path from a data source to another data source must exist to define and achieve data synchronization.

A web service middleware framework that provides an interface for external clients to enable them to access different local data sources with a transparent manner has been developed [19]. It has a module for configuring the middleware with the information of the heterogeneous data sources. As a new query is submitted to the middleware, it is routed based on the registered information at the middleware then the query is locally wrapped into different forms. In addition, the result of the query is combined into large XML dataset that is returned to the client who initiates the query.

Kester et al. succeeded to develop a system that integrates several drug stores, which are incorporated based on SOA concepts with web services [20]. The database systems of the drug stores have been incorporated via a service bus such that drugs can be queried from all registered geographically distributed data stores. The nearest geographical location of drug result can be monitored and tracked.

Each of the previously mentioned systems has its own desirable features, but all of them suffer from some limitations. Thus, we propose a new data integration platform called Resource Oriented Heterogeneous Data Integration Platform (ROHDIP) to overcome these limitations.

III. Methodology

The pre-integrated system design is shown in Fig. 1. which consists of: applications, network connections, and local databases. Applications are allocated on different machines that utilize different operating systems (Windows, UNIX, Linux…etc.). Each application is written in any programming language (e.g. C#, Java, JS, Ruby …etc.) A network is required for direct connection between each application and its corresponding local data source. This network can be LAN, MAN or WAN. There are several types of local databases each may have a specific data model with different database management systems (such as Relational, Object, Tree, Hierarchy, Flat file). Each application is able only to query its corresponding DBMS(Database Management System) that is installed on it.

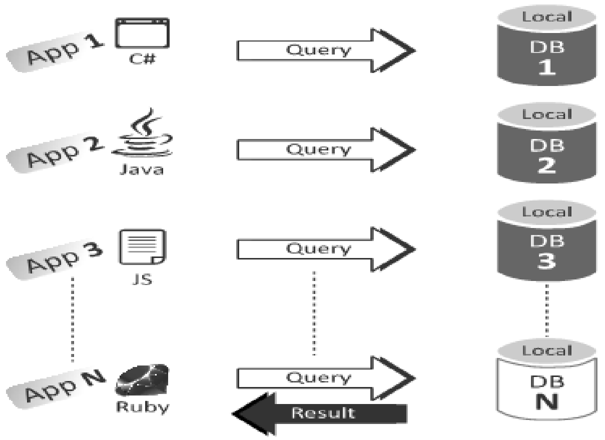


Fig. 1. Pre-Integration App(s)/Data source diagram

The previous architecture suffers from an obvious shortcoming that each application is restricted to query its local database only, to defeat this limitation the research attitude turned to focus on virtual integration approach.

The most challenging issue in the virtual data integration architecture is the network communication between the mediated schema and the data sources; see Fig. 2. This issue was solved by using Service-oriented architecture (SOA), as it exchanges data across the platform in the standard way through web services [21]. However, SOA data integration platforms still have some disadvantages [22] [23]. For example, size multiplication of the transmitted data leads to a negative impact on the network traffic and the system performance, especially when treating a large amount of data. Also, SOA platform suffers from higher latency and processing delay. Moreover, not all machines support the SOAP protocol (e.g. mobiles and embedded systems) as a native protocol. To overcome these limitations the Restful architecture is used.

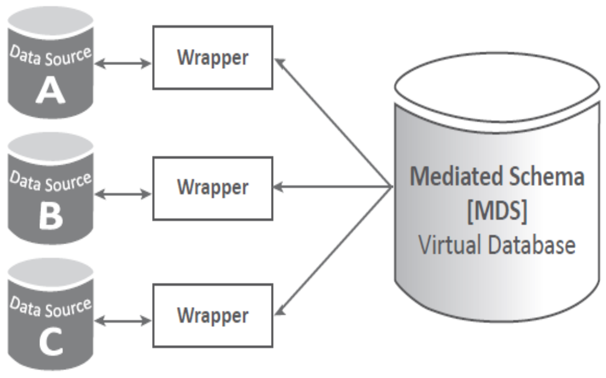


Fig. 2. Virtual data integration architecture

REST is a lightweight, easy and better alternative for the SOAP. Implementing the data exchanges across the platform using the Restful architecture of web service is more efficient in terms of both the network bandwidth utilization of the service requests transmitting over the Internet, as well as the latency incurred during these requests [22][24].

A. The Proposed Platform: ROHDIP

We propose a platform; namely, Resource Oriented Data Integration Platform (ROHDIP) that depends on the resource-oriented architecture (ROA) instead of the SOA architecture. ROA uses Representational State Transfer (RESTful) service based on HTTP protocol for communicating local data sources with mediated schema. ROHDIP is designed as a collection of collaborative RESTful resources allocated on distributed machines with different operating systems and constructed according to the ROA principles as shown in Fig. 3.

B. ROHDIP Architecture

The proposed platform architecture consists of three major steps: mediated schema creation, Data Source subscription, and mediated schema querying.

1) Mediated Schema Creation

TABLE I. MEDIATED SCHEMAS METADATA

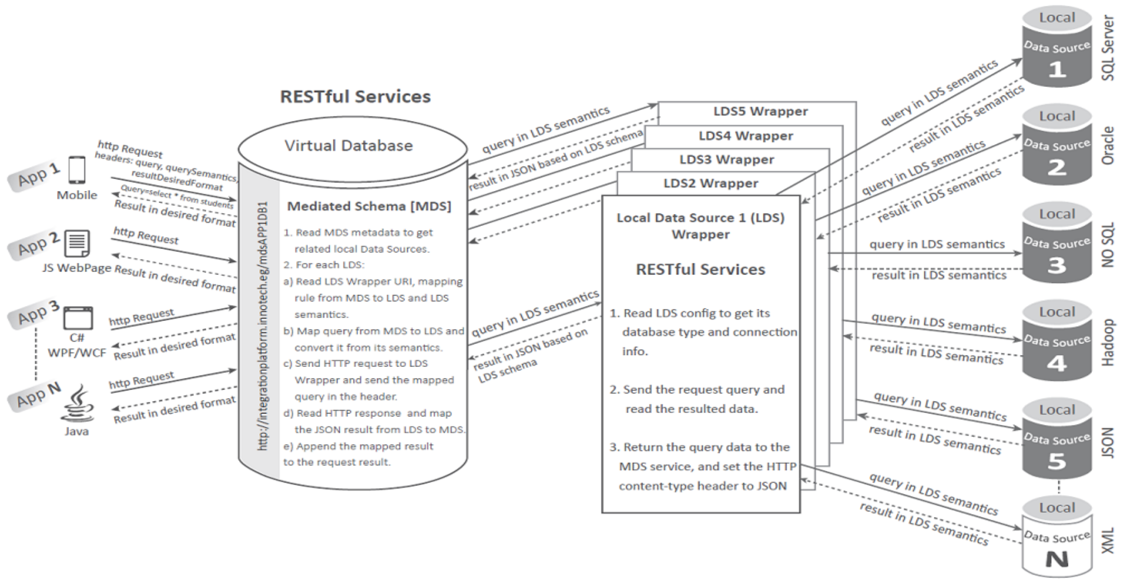
Mediated Schema ID	Mediated Schema Name	Schema Definition (JSON)	Subscribed Data Sources (JSON)
mdsStudents	StudentsVDB	{"StudentID":"","Name":"","Mobile":""}	[{"DataSourceId":"ds2", "DataSourceName":"Engineering"}]
mdsStaff	StaffVDB	{"StaffD":"","Name":"","Mobile":""}	{"DataSourceId":"ds3", "DataSourceName":"Commerce"}]
……….	………..	{…………..}	…………..
mdSchN	mdEmployess	{"EmployeeID":"","Name":"","Mobile":""}	[{"DataSourceId":"ds2", "DataSourceName":"Medicine"},[{"DataSourceId":"ds2","DataSourceName":"Engineering"}]

Every mediated schema "Virtual Database" has its metadata see TABLE I. The metadata contain: ID, name, corresponding schema definition in JSON (JavaScript Object Notation) format and a list of the subscribed data sources of the mediated schema in JSON format. Fig. 4. illustrates the mediated schemas metadata in JSON format.

2) Subscribed Data Sources

When a new data source needs to join the ROHDIP, it must be added to the subscribed data sources metadata; see TABLE II. The metadata contain ID, URI, name, data model/DBMS, connection information between the data source and its wrapper service in JSON format, schema definition in JSON format, wrapper schema transformation rules in JSON format and the result data format. The subscribed data sources metadata in JSON format are also illustrated in Fig. 4.

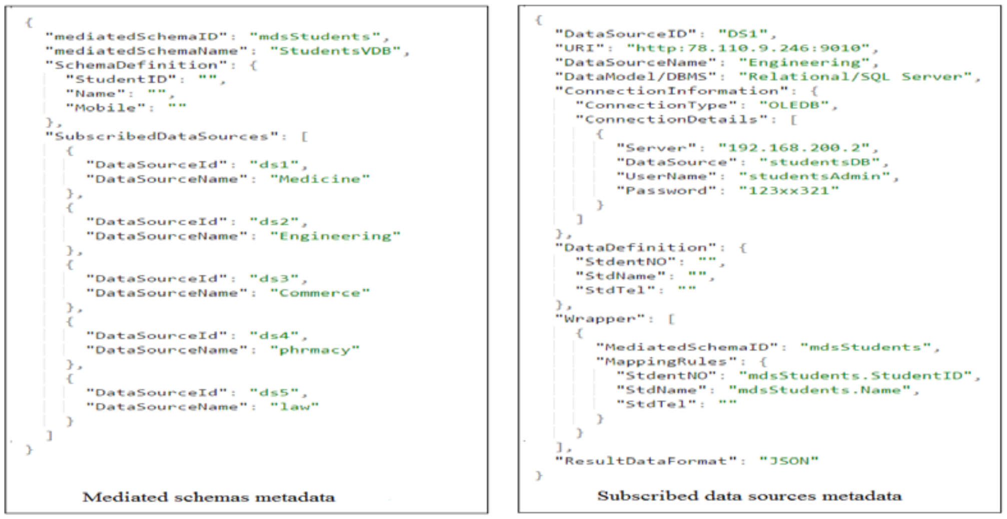
The wrapper schema transformation rules from the data source schema to the mediated schema (e.g. mdsStudents.StudentID = StdentNO, mdsStudents. StudentName = StdName, StdTel = null) are required as they enable the mediated schema to map the requested query to the data source schema semantics. Furthermore, the data source result format (e.g. JSON, XML, delimited text) is provided to enable the mediated schema to read the result and convert it to the requester desired format.



1. Read MDS metadata to get related local Data Sources.
2. For each LDS:
a) Read LDS Wrapper URI, mapping rule from MDS to LDS and LDS semantics.
b) Map query from MDS to LDS and convert it from its semantics.
c) Send HTTP request to LDS Wrapper and send the mapped query in the header.
d) Read HTTP response and map the JSON result from LDS to MDS.
e) Append the mapped result to the request result.

1. Read LDS config to get its database type and connection info.
2. Send the request query and read the resulted data.
3. Return the query data to the MDS service, and set the HTTP content-type header to JSON

Fig. 3. Proposed ROHDIP System Design



```
{
  "mediatedSchemaID": "mdsStudents",
  "mediatedSchemaName": "StudentsVDB",
  "SchemaDefinition": {
    "StudentID": "",
    "Name": "",
    "Mobile": ""
  },
  "SubscribedDataSources": [
    {
      "DataSourceId": "ds1",
      "DataSourceName": "Medicine"
    },
    {
      "DataSourceId": "ds2",
      "DataSourceName": "Engineering"
    },
    {
      "DataSourceId": "ds3",
      "DataSourceName": "Commerce"
    },
    {
      "DataSourceId": "ds4",
      "DataSourceName": "phrmacy"
    },
    {
      "DataSourceId": "ds5",
      "DataSourceName": "law"
    }
  ]
}
```

Mediated schemas metadata

```
{
  "DataSourceID": "DS1",
  "URI": "http:78.110.9.246:9010",
  "DataSourceName": "Engineering",
  "DataModel/DBMS": "Relational/SQL Server",
  "ConnectionInformation": {
    "ConnectionType": "OLEDB",
    "ConnectionDetails": [
      {
        "Server": "192.168.200.2",
        "DataSource": "studentsDB",
        "UserName": "studentsAdmin",
        "Password": "123xx321"
      }
    ]
  },
  "DataDefinition": {
    "StdentNO": "",
    "StdName": "",
    "StdTel": ""
  },
  "Wrapper": [
    {
      "MediatedSchemaID": "mdsStudents",
      "MappingRules": {
        "StdentNO": "mdsStudents.StudentID",
        "StdName": "mdsStudents.Name",
        "StdTel": ""
      }
    }
  ],
  "ResultDataFormat": "JSON"
}
```

Subscribed data sources metadata

Fig. 4. Mediated schemas, Subscribed data sources metadata in JSON format

TABLE II. SUBSCRIBED DATA SOURCES METADATA

DataSourceID	URI	DataSource Name	DataModel/ DBMS	Connection-Information	DataDefinition	Wrapper	ResultDataFormat
DN1	http://78.110.9.246:9010	Medicine	Relational/SQL Server	{"ConnectionType":"OLEDB","ConnectionDetails":[{"Server":"192.168.200.2","DataSource":"studentsDB","UserName":"studentsAdmin","Password":"123xx321"}]}	{"StdentNO":"","StdName":"","StdTel":""}	[{"MediatedSchemaID":"mdsStudents","MappingRules":{"StdentNO":"mdsStudents.StudentID","StdName":"mdsStudents.Name","StdTel":""}}]	JSON
DN2	http://78.110.9.246:9050	Commerce	XML	{………….}	{……….}	[{………….}]	XML
DNm	http://78.110.9.246:9090	Engineering	JSON	{………….}	{……….}	[{………….}]	CSV

3) Mediated Schema "Virtual Database" Query

In order to query the mediated schema from any location and from any querying device, using the HTTP verb "GET", we need to send HTTP request to the mediated schema URI i.e. http://IntegrationPlatfrom.Innotech.com.eg, in which we have to fill the requested mediated schema ID HTTP header i.e. "mdsStudents" and feed the "query" HTTP header with the desired query string i.e. "select * from mdsStudents". Upon receiving the HTTP request; the mediated schema RESTful service will check if the mediated schema ID is valid by examining the mediated schemas metadata.

If the mediated schema is valid, the mediated schema RESTful service will iterate through all its corresponding subscribed data sources by getting them from the mediated schemas metadata. Then, it will read its data model/DBMS, connection information, data definition, and wrapper details "mapping rules". After that, HTTP request will be sent to all the subscribed data sources wrapper services URI(s) to retrieve the HTTP request result and append it to the mediated schema query result JSON data.

Finally, convert the JSON result consolidated from all the data sources to the requester data format then return it back to the requester in the body section of the requester query HTTP response.

IV. Results and Discussions

The proposed platform is evaluated using the KDD'99 dataset, which includes 41 features extracted from DARPA (Defense Advanced Research Projects Agency) TCP dump in 1998 [25] [7]. The kdd'99 dataset consists of 494,021 connection records. We divide the KDD'99 dataset into six groups of smaller n-record datasets where n equals 5, 50, 500, 1000, 5000 and 10,000 records. The generated datasets are distributed over three servers each of which are of type PowerEdge R220 Rack Server, processor Xeon CPU e3-1220 v3 3.1GHZ, and Ram 24 GB.

We compare the performance of our proposed platform with SOA data integration framework, in terms of the end-to-end response time each query takes to retrieve a different number of rows. The response time is estimated for 5000, 25000, 50000, 75000, 100000 and 125000 rows. The proposed platform outperforms SOA, considering all the mentioned retrieved data sizes.

Fig. 5 through 10 illustrate the significant performance progress of the proposed ROHDIP platform comparing to the SOA framework regarding different sizes of retrieved query result. ROHDIP achieves the required integration of data retrieved from a query with minimum response time compared to SOA among a different number of data sets. The results clarify that the gap between ROA and SOA increases as the query result size increases. The results demonstrate that ROA is better than SOA in the data integration field.

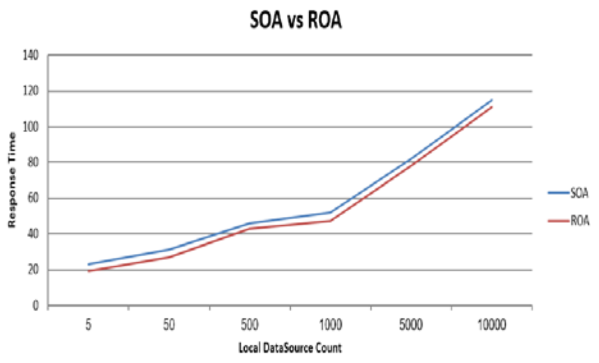


Fig. 5. Response time of ROHDIP vs. SOA for 5000 rows as a query result

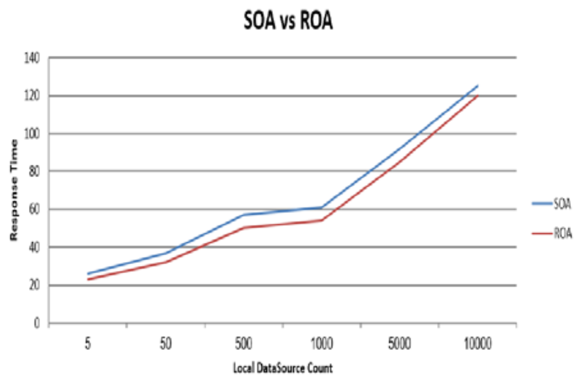


Fig. 6. Response time of ROHDIP vs. SOA for 25000 rows as a query result

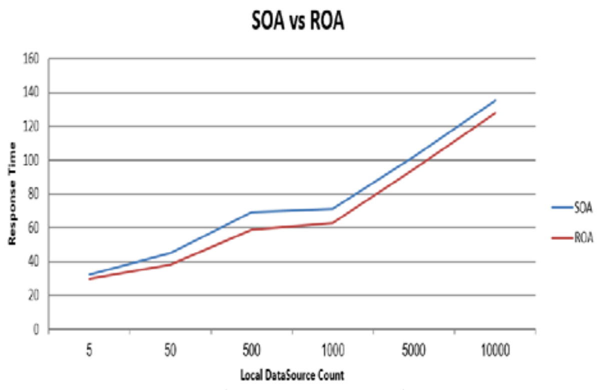


Fig. 7. Response time of ROHDIP vs. SOA for 50000 rows as query a result

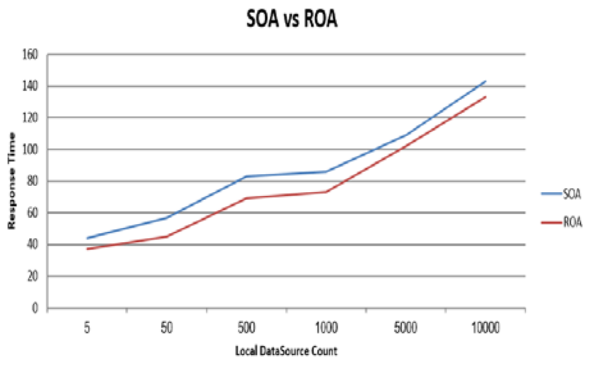


Fig. 8. Response time of ROHDIP vs. SOA for 75000 rows as a query result

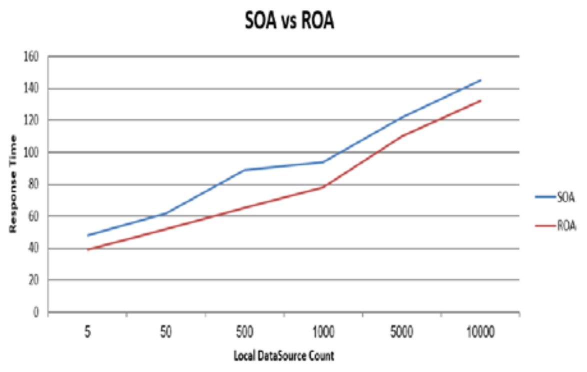


Fig. 9. Response time of ROHDIP vs. SOA for 100000 rows as a query result

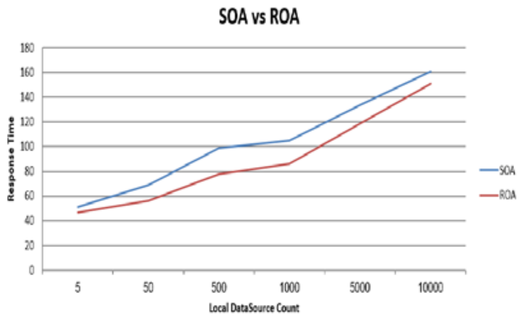


Fig. 10. Response time of ROHDIP vs. SOA for 125000 rows as a query result

V. Conclusion and Future Work

Data integration is considered as the most urgent data task due to the daily increasing of data in heterogeneous data sources. In this paper, Resource Oriented Heterogeneous Data Integration Platform (ROHDIP) is proposed in order to integrate data from multiple heterogeneous data sources providing a unified query interface. The results evidence that ROA outperforms SOA for any query result size on a variety of distributed data sources achieving the minimum response time.

We believe that the vision and research contribution described in this paper will serve large-scale data gathering and integration studies in the near future.

As mentioned in the paper, the heterogeneous data sources are distributed and allocated on different machines, so, our future vision is to apply parallel processing or parallel querying between the mediated schema RESTful service and the wrappers RESTful services. Further investigations are needed concerning the security issues.

Acknowledgment

This paper is supported by iNNOTECH development corporate which provided us with the suitable devices for this research.

References

[1] Y. Liu and M. Xia, “Research of heterogeneous database integration based on XML,” ICMET 2010 - 2010 Int. Conf. Mech. Electr. Technol. Proc., pp. 793–796, 2010.

[2] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati, “Information Integration: Conceptual modeling and reasoning support,” 3rd IFCIS Int. Conf. Coop. Inf. Syst., pp. 280–289, 1998.

[3] A. Y. Levy, “Logic-based techniques in data integration,” Logic-based Artif. Intell., pp. 575–595, 2000.

[4] J. A. R. Castillo, A. Silvescu, D. Caragea, J. Pathak, and V. G. Honavar, “Information extraction and integration from heterogeneous, distributed, autonomous information sources - A federated ontology-driven query-centric approach,” Proc. 2003 IEEE Int. Conf. Inf. Reuse Integr. IRI 2003, pp. 183–191, 2003.

[5] J. A. Reinoso-castillo, “Ontology-driven information extraction and integration from heterogeneous distributed autonomous data sources : A federated query centric approach,” Architecture, 2002.

[6] P. Ziegler and K. R. Dittrich, “Three Decades of Data Integration — All Problems Solved? ” 18th IFIP World Comput. Congr. (WCC 2004), vol. 12, pp. 3–12, 2004.

[7] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “The KDD process for extracting useful knowledge from volumes of data,” Commun. ACM, vol. 39, no. 11, pp. 27–34, 1996.

[8] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner, “Ontology-based integration of information-a survey of existing approaches,” IJCAI Work. Ontol. Inf. Shar., pp. 108–117, 2001.

[9] S. Abiteboul, O. Benjelloun, and T. Milo, “Web services and data integration,” Proc. Third Int. Conf. Web Inf. Syst. Eng. 2002. WISE 2002., pp. 3–6, 2002.

[10] G. Elsheikh, M. Y. Elnainay, S. Elshehaby, and M. S. Abougabal, “SODIM: Service Oriented Data Integration based on MapReduce,” Alexandria Eng. J., vol. 52, no. 3, pp. 313–318, 2013.

[11] F. Zhu, M. Turner, I. Kotsiopoulos, K. Bennett, M. Russell, D. Budgen, P. Brereton, J. Keane, P. Layzell, M. Rigby, and J. Xu, “Dynamic Data Integration Using Web Services,” Int. Conf. Web Serv., 2004.

[12] H. Garcia-Molina and others, “The {TSIMMIS} Approach to Mediation: Data Models and Languages,” J. Intell. Inf. Syst., vol. 8, no. 2, pp. 117–132, 1997.

[13] C. Batini, M. Lenzerini, and S. B. Navathe, “A comparative analysis of methodologies for database schema integration,” ACM Comput. Surv., vol. 18, no. 4, pp. 323–364, 1986.

[14] A. Y. Levy, A. Rajaraman, and J. J. Ordille, “Querying Heterogeneous Information Sources Using Source Descriptions,” Proc. 22th Int. Conf. Very Large Data Bases, vol. 1, pp. 1–26, 1996.

[15] S. Sathya and M. Victor Jose, “Application of Hadoop MapReduce technique to Virtual Database system design,” 2011 Int. Conf. Emerg. Trends Electr. Comput. Technol. ICETECT 2011, pp. 892–896, 2011.

[16] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. M. S. de Souza, and V. Trifa, “SOA-Based Integration of the Internet of Things in Enterprise Services,” pp. 968–975, 2009.

[17] A. L. Sanz, M. N. García, and V. F. Batista, “XML based integration of web, mobile and desktop components in a service oriented architecture,” Adv. Soft Comput., vol. 50, p. 565, 2009.

[18] S. B. Li, Y. Hu, and Q. S. Xie, “Heterogeneous System Integration Based on Service Component,” Appl. Mech. Mater., vol. 20–23, pp. 1305–1310, 2010.

[19] X. Wei, “Heterogeneous Database Integration Middleware Based on Web Services,” Phys. Procedia, vol. 24, pp. 877–882, 2012.

[20] Q. Kester and A. I. Kayode, “Using SOA with Web Services for effective data integration of Enterprise Pharmaceutical Information Systems,” pp. 1–8.

[21] P. Version, “Mumbaikar, S., & Padiya, P. (2013). Web services based on soap and rest principles. International Journal of Scientific and Research Publications, 3(5). Chicago,” Int. J. Sci. Res. Publ. 3(5). Chicago, vol. 3, no. 5, 2013.

[22] G. Mulligan and D. Gračanin, “A comparison of soap and rest implementations of a service based interaction independence middleware framework,” Proc. - Winter Simul. Conf., pp. 1423–1432, 2009.

[23] K. P. Pavan, A. Sanjay, and P. Zornitza, “Comparing Performance of Web Service Interaction Styles : SOAP vs. REST,” 2012 Proc. Conf. Inf. Syst. Appl. Res., pp. 1–24, 2012.

[24] H. Hamad, M. Saad, and R. Abed, “Performance evaluation of restful web services for mobile devices,” Int. Arab J. e-Technology, vol. 1, no. 3, pp. 72–78, 2010.

[25] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, “The 1999 DARPA o -line intrusion detection evaluation,” Comput. Networks, vol. 34, no. 4, pp. 579–595, 2000.