International Journal of Innovative Computing, Information and Control
Volume **8**, Number **11**, November **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 7819–7828

DSM: A DATA SERVICE MIDDLEWARE FOR SHARING DATA IN PEER-TO-PEER COMPUTING ENVIRONMENTS

Sofien Gannouni, Mutaz Beraka and Hassan Mathkour

Computer Science Department
College of Computer and Information Sciences
King Saud University
Riyadh 11451, Kingdom of Saudi Arabia
{ gnnosf; mathkour }@ksu.edu.sa; mutaz999@hotmail.com

Received July 2011; revised February 2012

Abstract. *The advances in network computing models evolve the way people share their resources. The file sharing is a peer-to-peer (P2P) computing model that allows users to share and exchange files. However, it does not meet the growing need of sharing existing widespread data-sources. During the last decade, many research and development efforts have been deployed pertaining to proposing approaches for accessing remote, heterogeneous and autonomous data sources. We propose a middleware that allows non-expert users to share and integrate their data. Data are export and deployed as services. As such, they are easily discovered, uniformly accessible using standard SOAP requests and would be integrated through service composition. This paper focuses on the description of the architecture of the proposed middleware.*

Keywords: Data sharing, Data service approach, Data service layer, UDDI registry, Service composition

1. **Introduction.** Sharing and integrating existing autonomous, distributed and heterogeneous data sources allow companies and individuals to gain a holistic understanding of data. They have been recognized as of a great importance to small and huge-scale businesses. Enhancing the accessibility and the reusability of these data entails the development of new approaches for data sharing. In the literature, different data sharing approaches have been investigated and applied in different computing environments. These approaches vary in terms of concepts and technology standards. The most widely known data sharing approaches are transaction processing monitor [1,2], tuplespace [2,3], resource description framework [2,4] and data service layer (DSL) [2,5].

Service-oriented computing has emerged as the eminent distributed computing model for developing reusable loosely coupled service-centric business applications [6,7]. DSL provides a uniform view of the data in an SOA-based system. It is responsible for accessing structured, semi-structured and un-structured data sources using Web services or representational state transfer (REST) style Web services. The main advantage of this approach is that it reduces the complexity of developing new applications that integrate data from several data sources [5,8-10].

As the data service layer is a key factor for a successful development of SOA-based systems various DSLs propose different mechanisms for achieving efficient data access. Most of the existing DSLs, e.g., [11,12], are dedicated to single site users and do not satisfy the need of users to access efficiently data at different locations. Some prototypes, e.g., [13,14], develop solutions for efficient distributed data access, but they do not consider their users as peers. Implementations of DSL prototypes for Peer to Peer (P2P) computing environment are rare and almost nonexistent. In addition, existing systems, e.g., [15], do not provide a comprehensive and complete solution for P2P data sharing.

In this paper we present a simple yet efficient data service layer that provides access to heterogeneous data sources. We propose to export Databases, Excel files, XML files, CSV files as Services as a possibility to tackle data sharing and integration problems. Furthermore, we propose an infrastructure that allows peers to customize, compose, and deploy complex data sources.

The remainder of this paper is structured as follows. Section 2 describes our motivations to propose and develop a Data Service Middleware (DSM). Section 3 reviews the existing approaches for remote data sources access and highlights some prototypes that adopt the Date Service approach. Section 4 describes the architecture of DSM as well as its main components of DSM. Section 5 summarizes this and highlights future improvements of the proposed middleware.

2. Motivations.

When we are looking back at the computer industry, we can clearly identify the growing need of data sources in small and large scale business. As per a recent survey done by the Ponemon Institute, 90 percent of organizations reported having more than 100 databases and 23 percent have more than 1,000 [16]. This massive presence of databases in these organizations is due to the fact that many of the employees of these organizations have created their own "databases" in response to the requirement of the tasks they are responsible for. These people require often integrating and sharing their data sources to gain the holistic understanding of the whole organization's data.

We propose to export every data source as a Web-service, called a Data-service, which contains a set operations (capabilities) generated based on the analysis of the data source schema. The invocation of the operations of a Data-service will lead to the execution of appropriate data manipulation statements on the corresponding data source. In order to highlight the benefits of this approach, we discuss the following motivation sample.

Consider three data sources namely a beekeeping database (BK), a fauna and flora data source (FF) and a climatic data source (CL). The BK data source contains information about hives and bee colonies (health, species, apiaries production, etc.). The FF data source provides information about the different types of vegetation of various regions. The CL data source provides information about climatic prediction (temperature, humidity, etc.). Exporting these data sources as Dataservices will provide uniform access to the data they store. Thus, the heterogeneity and the location of the data sources become transparent to the users and retrieving data from these data sources becomes a simple invocation of the operations of the Data-services. Moreover, the integration of the existing heterogeneous data sources could be obtained simply through service composition. Indeed, a beekeeper may compose new Data-service that aggregates capabilities of the BK's corresponding Data-service and capabilities of the FF's corresponding Data-service. The composite Data-service allows the beekeeper to optimize his production by identifying areas of overgrazing with potential seasonal bee flora interest. The beekeeper may also compose a new Data-service that aggregates capabilities of BK, FF and CL data sources' corresponding Data-services. The new composite Data-service provides useful data that would help beekeeper in (dis) placement of hives according to Botanico-climatic conditions of the moment.

The underpinning for an organization's use of the proposed approach is the ability to discover existing data sources, to have a uniform access to the data sources and to save time in the development of new business applications by enabling the integration of existing data sources through service composition. [9] reported that up to 70 percent of the time spent to develop applications that integrate data from different data sources is consecrate to accessing distributed data.

3. **Related Work.**

During the last decade, much research and development effort has been put into proposed approaches for accessing remote, heterogeneous and autonomous data sources. In our review of the literature, we identify the following approaches: Transaction Processing Monitors, Tuplespace, Resource Description Framework and Data Service.

3.1. **Transaction processing monitors.** A TPM provides an infrastructure for building and administering complex transaction processing systems with a large number of clients and multiple servers [2,17]. It supports mainly services for submitting user queries, routing them through servers for processing, coordinating the two-phase commit when the transactions are running over multiple servers and ensuring that each transaction satisfies the Atomicity, Consistency, Isolation, Durability (ACID) prosperities [1,2]. These properties guarantee the database's consistency over time and guard against hardware and software errors [2].

3.2. **Tuplespace.** Tuplespace was initially proposed to support the Linda parallel programming language [18,19], which was developed by David Gelernter and Nicholas Carriero at Yale University [2]. It provides a set of primitive operations to insert, fetch and retrieve data from a shared space that stores user data [2]. It may be considered a form of distributed shared memory which allows the data providers to post their data as tuples in the shared space, and the data consumers to fetch and retrieve data which matches a certain pattern from that space [2].

3.3. **Resource description framework.** RDF is a Semantic Web technology that supports the exchange of data and knowledge on the Web [2]. It is a standard format developed by W3C for representing and storing any kind of data as Web resources on the Web [4,20]. In practice, RDF resources are identified by Uniform Resource Identifiers (URIs) on the Web [21]. This URI reference is formed by a URI namespace and a local name [3,21].

3.4. **Data service approach.** The Data Service Approach is the most widely used approach nowadays for data exchange. It embodies the Service-Oriented Architecture (SOA) principles to expose data stored in heterogeneous and autonomous data sources [2]. It supplies a Data Service Layer (DSL) as a mechanism for masking heterogeneity between data sources such as databases, files or spreadsheets, and make them available as Web services or as set of Representational State Transfer (REST) style Web services. The main advantage of this approach is that it reduces the complexity of developing new applications that integrate data from several data sources [5,9,10]. The following prototypes adopt the data service approach:

- The AquaLogic Data Service Platform (ALDSP) is dedicated for the design and maintenance of a data service layer [11]. It provides a declarative technique to create and implement data services for composite applications within an enterprise.
- DBProxy [13] is a data replication system which maintains and replicates business application data over the network. In DBProxy, both client queries and business back-end data are partially cached in one or more proxies. Client queries can be executed on the local database of the server.
- ESDS [14] is a distributed data service scheme called Edge Server Data Service which integrates DSL and network data management technologies together to allow users at different locations to access data efficiently.

- WSO2 [15] is software built on top of WSO2 Carbon, a lightweight high-performance platform for taking data stored in data sources and making them available as Web services. It uses Axis2 as the underlying SOAP processing engine [15]. WSO2 supports relational databases, CSV files and Microsoft Excel files. In addition, WSO2 supports security using WS-security standards and reliability using WS-Reliable Messaging standard [15].

4. **Overview of the Main Architecture of DSM.** DSM is a Service-Oriented Middleware that embodies the principles of SOA for sharing data in a P2P environment. It allows peers to export their data as services and to have access to those of others using the data services they publish. It provides a set of rich and easy to use services allowing non-expert users to share their data with each other. Moreover, it offers a semi-dynamic service composition engine allowing users to integrate data from different resources by composing new data services.

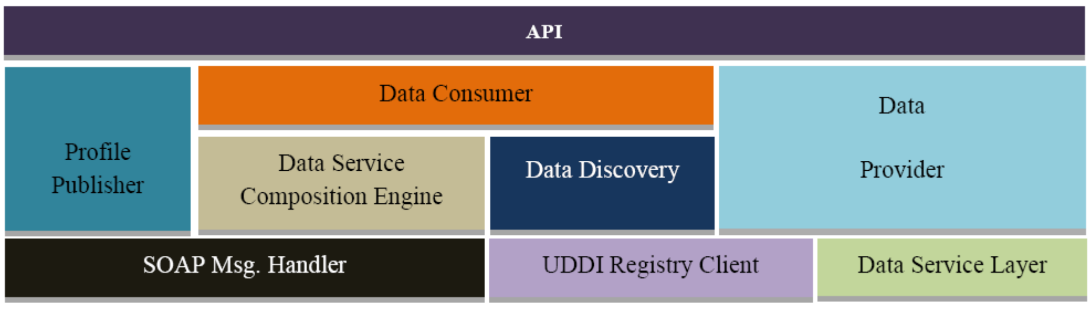

FIGURE 1. Main architecture of DSM

As shown in Figure 1, the main architecture of DSM consists of the following components:

- ***Profile Publisher (PP)***: PP is responsible for creating and updating the profile of a peer. Only users having profiles are allowed to publish and/or use data services.
- ***Data Provider (DP)***: DP is responsible for exposing (exporting) users' data sources as data services and publishing them in a UDDI registry.
- ***Data Discovery (DD)***: DD is responsible for discovering data services based on user's search criteria.
- ***Data Consumer (DC)***: DC is responsible for invoking the discovered data-services which provide access to their corresponding back-end remote data sources.
- ***Data Service Layer (DSL)***: DSL is responsible for providing access to the local data sources (excel files, XML data, relational databases, etc.).
- ***Data Service Composition Engine (DSCE)***: DSCE is responsible for generating and describing a new business process that integrates existing endpoints into a new endpoint. It is also responsible for interpreting the business logic of the process to get the XML result of the invocation.
- ***UDDI Registry Client***: It provides access to basic UDDI functionalities.
- ***SOAP Msg Handler (SMH)***: SMH is responsible for reading and writing SOAP messages, sending and receiving these messages. Moreover, it is responsible for parsing SOAP XML responses to extract embedded data.

4.1. **Data service layer (DSL).** The DSL allows exporting part-of or the whole user's data sources. It generates new data-services based on the scheme of the user's data sources It preserves the local data sources' autonomy of design, association and execution.

As depicted in Figure 2, the DSL consists mainly of the three following sub-components:

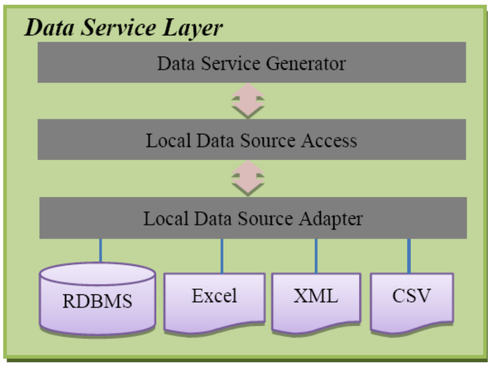

FIGURE 2. Architecture of DSL

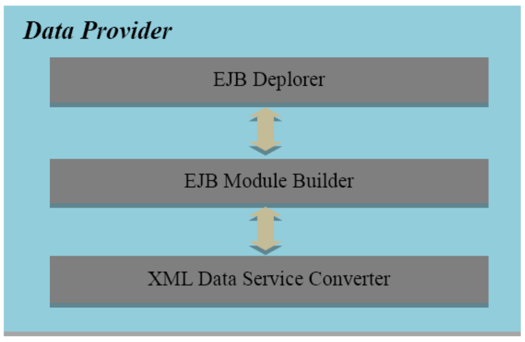

FIGURE 3. Architecture of the data provider component

- ***Local Data Source Access (LDSAccess)***: LDSAccess is responsible for providing uniform access to heterogeneous data sources. It allows discovering the metadata of the user's data sources and retrieving data from them.
- ***Local Data Source Adapter (LDSAdapter)***: LDSAdapter is responsible for translating the meta-data discovery requests and the data retrieve queries submitted by the LDSAccess into statements appropriate to the local data sources.
- ***Data Service Generator (DSG)***: DSG is responsible for generating a new data-service based on the schema of the user's data source. Firstly, it translates the data source's schema into an XML format. Next, it parses the XML format of the schema and generates a Java class which contains a set of appropriate operations that provide access to the data source tables and columns. Finally, it uses Web services and EJB annotations to annotate the generated Java class.

4.2. **Data provider (DP).** The DP component is responsible for deploying the data service generated by the DSL under the application server and publishing its description in a UDDI registry. DP performs the following tasks to expose a data source as a data service:

(1) Call the DSL adapter to create a new data service class that describes the data source.
(2) Prepare XML data-service descriptor which contains information about the generated data service such as the service name, its description and useful binding information.
(3) Generate a new EJB module which includes the necessary files and artifacts for this service.
(4) Deploys the generated EJB module under the application server.
(5) Publish the data service through the UDDI registry using the UDDI registry client.
(6) Write information about the status of the deployment and publishing processes into a log file.

As shown in Figure 3, DP consists of the following three sub-components:

- ***XML Data Service Converter***: It is responsible for converting unstructured information into an agreed XML format. This format allows other components to parse data easily in a structured manner.
- ***EJB Module Builder***: It is responsible for generating a new EJB module with the necessary files and artifacts.
- ***EJB Deplorer***: It is responsible for deploying (un-deploying) the generated EJB module under the application server using Ant-API.

4.3. **UDDI registry client (URC).** URC component is responsible for accessing any UDDI v3 compliant server using a valid security token (publisher profile). It allows peers to use their publisher profiles (username and password) to create, update and delete business entities, to publish and/or remove data services under a specific business entity, and to discover data services that are published by other peers.

4.4. **SOAP Meg. handler (SMH).** SMH is responsible for reading and writing SOAP messages. It sends and receives these messages through the Internet using SOAP with Attachments API for Java (SAAJ). Moreover, it parses SOAP XML responses to extract data. Figure 4 presents the different sub-components of SMH.

4.5. **Data discovery (DD).** The Data Discovery (DD) component is responsible for discovering the data-services published by the other peers. It uses the UDDI registry client to retrieve the data services' descriptors according to the user's criteria and values.

As shown in Figure 5, DD consists of the following three sub-components:

- ***Request Builder***: This is responsible for building a search request based on the user's criteria and querying the data services repository (UDDI Registry) using URC. The result is a list of XML data-services' descriptors.
- ***UDDI Registry Client Connector***: This is responsible for sending the request to the UDDI Registry Client and for receiving the result.
- ***XML Data Service Parser***: This is responsible for parsing the XML data-services' descriptors to retrieve the information about each data service, such as the name, the description and the URL of the WSDL document.

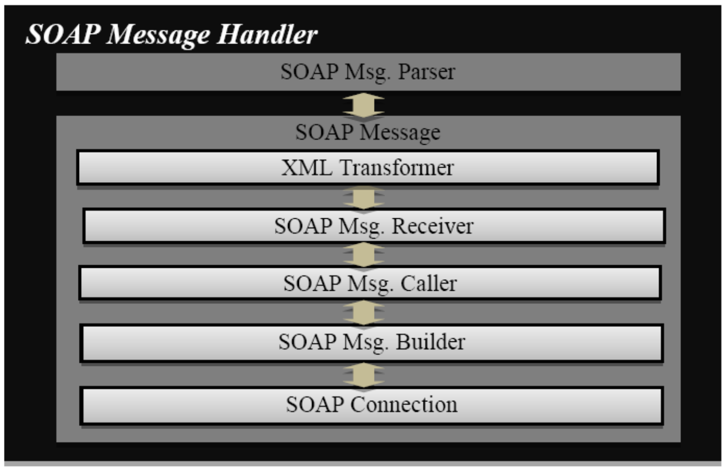

FIGURE 4. Architecture of SOAP message handler

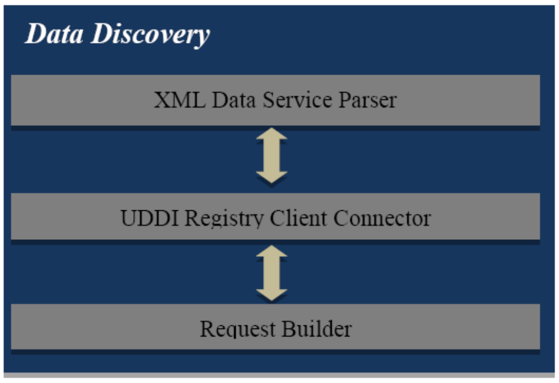

FIGURE 5. Architecture of DD

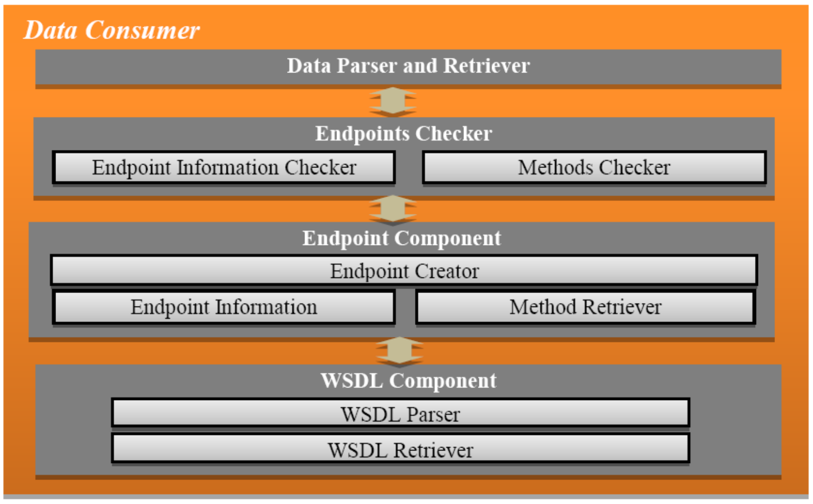

FIGURE 6. Architecture of DC

4.6. **Data consumer (DC).** The DC component is responsible for enabling access to the back-end remote data sources through the invocation of the operations of the published data-services. Firstly, DC defines and submits the user's search criteria to the Data Discovery component. It receives back and parses the result of the discovery phase. Further, it allows the user to specify the operations to invoke and adds them to a remote invocation list. Finally, DC passes this latter list to the data service composition engine for processing and parses the result returned back. DC parses the XML results and returns back the data to the user.

As depicted in Figure 6, DC consists of the following six sub-components:

- ***WSDL Component***: It is responsible for retrieving and parsing the WSDL (data-service descriptor) document associated to the data service.
- ***Endpoint Component***: This is responsible for extracting useful information from the parsed document and for creating a new endpoint object.
- ***Endpoint Checker***: It is responsible for checking the completion and the correctness of the information on an endpoint as well as the correctness of selected methods.
- ***Data Parser and Retriever***: This is responsible for parsing the XML results returned back by the DSCE engine.

4.7. **Data-service composition engine (DSCE).** DSCE is responsible for composing new data-services from capabilities (operations) of existing data-services by providing a description of their corresponding business process. It is also responsible for parsing, interpreting and supplying the result of a business process description.

As described in Figure 7, DSCE consists of the two following main components:

- ***Business Process Generator (BPG)***: It is responsible for composing a new data-service as an aggregation of a set of existing data services. It generates for the composite data-service a new business process, written in Data-Service Composition Language (DSCL) which is derived from BPEL, based on the list of selected endpoints, target methods within these endpoints, execution constraints and invocation options.
- ***Business Process Interpreter (BPI)***: It is responsible for parsing, interpreting and executing the logic of a DSCL business process to perform remote method invocation sequentially or in parallel based on the execution mode and activities precedence graph. BPI consists of the following sub-components:

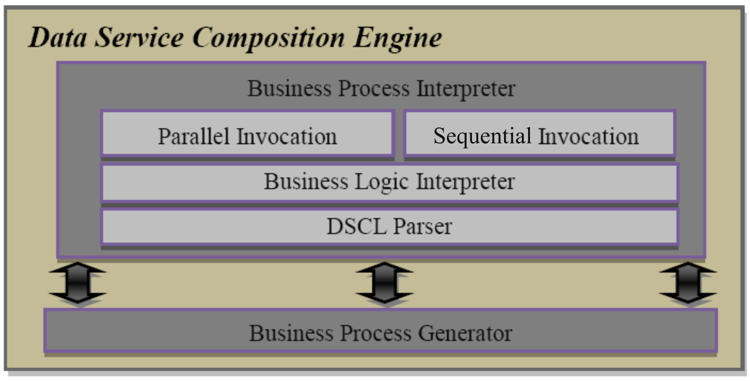
Data Service Composition Engine
Business Process Interpreter
Parallel Invocation
Sequential Invocation
Business Logic Interpreter
DSCL Parser
Business Process Generator

Figure 7. Architecture of DSCE

– ***DSCL Parser***: It is responsible for parsing the XML description of a DSCL business process to extract useful information about the business process. It does not supply information on the logic of the business process. Based on the extracted information, it creates a process information object.
– ***Business Logic Interpreter***: It is responsible for interpreting and executing the logic of the business process. This sub-component uses the process information object generated by the parser to get the required information to perform the invoke activity such the method name, input parameters, output parameters and messages of that method.
– ***Sequential Invocation***: It offers the ability to perform a set of invocation activities of a business process in a sequential mode.
– ***Parallel Invocation***: It provides the ability to perform a set of invocation activities of a business process in a parallel manner using the Fork-Join capabilities of Java.

4.8. **Profile publisher (PP).** To enforce proper access to the UDDI registry and prevent it from non-authorized access, we provide an authentication system, which is a Web service integrated with jUDDI. We implement a suitable authentication mechanism that meets our requirement of having a valid authentication token for each request sent to jUDDI. Obtaining this token requires the correct credentials. However, the Profile Publisher is a Web-service client that is responsible for performing the following operations:

- Register a new peer in the UDDI registry by creating a new profile.
- Check that the credentials of the peer profile publisher are correct before allowing the peer to perform operations such as publishing and inquiring.
- Modify profile information such as the password and email or IP address.

5. **Conclusions and Future Work.** We have proposed and successfully developed a novel middleware named Data Service Middleware (DSM) that enables users to share their data sources in a P2P environment. It relies on a service-oriented approach to export users' data sources as data-services, discover and invoke those services. It also relies on a process-oriented approach to provide service composition capabilities in order to support virtual data integration. The underpinning for an organization's use of the proposed middleware is the ability to discover existing data sources, to have a uniform access to them regardless their heterogeneity and their location and to save time in the development of new business applications by enabling the integration of existing data sources through service composition.

DSM consists of three main components: the Data-Provider, the Data-Discovery and the Data-Consumer. The Data Provider enables the users to export and publish their data sources as a data-services in a UDDI registry. The Data Discovery component allows the peers to discover published data-services. The Data Consumer enables the peer to invoke operations of the discovered services. The invocation of an operation of data-service will lead to the execution of appropriate data manipulation statement on the corresponding back-end data source. Moreover, the Data consumer allows the users to integrate (virtual integration) data from heterogeneous data sources by enabling the user to compose new data-services that aggregates operations of different data-services. The execution of the composite data-services could be done in sequential or in parallel mode.

DSM solves the heterogeneity between data sources by implementing an abstract data layer called DSL which provides uniform access to the data sources. Furthermore, it adopts a standard platform-independent technology (Web-services technology) to export those data sources as data-services. DSM meets the current demands of data sharing in a P2P environment by providing a set of well-defined, ready-made and easy-to-use services that allow non-expert users to publish, discover and use data-services without writing any additional code and with less effort.

For the time being, we assume that the schemes of the data sources are stable and do not change. Therefore, if changes are made to these schemes, the corresponding data-services are no more appropriate and require to be updated. Changing these services may cause some peers to crash. We intend to support a multi-versioning system that ensures service availability for peers, who already derived new services from those updated ones. We intend also to introduce a caching mechanism into DSM in order to reduce the execution time of users' requests and increase the data availability when back-end data sources experience some deficiencies.

Acknowledgment. This work was supported by the Research Center, College of Computer and Information Sciences, King Saud University. The authors are grateful for this support. They also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] G. Alonso, *Transaction Processing Monitors*, Computer Science Department, Swiss Federal Institute of Technology (ETHZ), http://masteritgov.dia.uniroma3.it/didattica/MW/ Middleware-04-TP-Monitors.ppt, 2011.

[2] S. Gannouni, H. Mathkour and M. Beraka, A comparative survey of data sharing approaches and their applications in distributed computing environments, *Journal of Theoretical and Applied Information Technology*, vol.33, no.1, pp.42-57, 2011.

[3] S. Capizzi, *A Tuple Space Implementation for Large-Scale Infrastructures*, Ph.D. Thesis, Universit'a di Bologna, Padova, 2008.

[4] N. Alexander and S. Ravada, RDF object type and reification in the database, *Proc. of the 22nd International Conf. on Data Engineering*, pp.93-103, 2006.

[5] J. Bloomberg and J. Goodson, Best practices for SOA: Building a data service layer, *SOA World Magazine*, vol.8, no.5, 2008.

[6] M.-H. Lin, H.-Y. Kung, W.-K. Lai, Y.-H. Lan and C.-S. Shieh, A SOAP-based domain web service middleware: Design and implementation, *ICIC Express Letters, Part B: Applications*, vol.2, no.2, pp.279-286, 2011.

[7] C.-C. Chang and C.-D. Tsai, Web service aggregation of cloud computing: An international journal of research and surveys, *ICIC Express Letters, Part B: Applications*, vol.2, no.3, pp.711-715, 2011.

[8] R. Seeley, *SOA Principles Apply to Data Access and Management*, http://www.SearchWeb Services.com, 2007.

[9] K. Goundar, S. Singh and X. Ye, An investigation into concurrency control mechanisms in data service layers, *Proc. of the 14th Asia-Pacific Software Engineering Conference*, 2007.

[10] M. Nikoo, *The Data Layer – Build or Buy?* Dunstan Thomas Consulting, 2003.

[11] M. Carey, Data delivery in a service oriented world: The BEA AquaLogic data services platform, *Proc. of the 2006 ACM SIGMOD International Conference on Management of Data*, Chicago, IL, USA, pp.695-705, 2006.

[12] *Composite Software Composite Data Services Architecture*, http://www.compositesw.com/index.php/products/compositeapplication-data-service, 2012.

[13] K. Amiri, S. Park and R. Tewari, A self-managing data cache for edge-of-network web applications, *Proc. of the 11th International Conference on Information and Knowledge Management*, McLean, pp.177-185, 2002.

[14] R. Yin and X. Ye, An efficient data service layer, *Proc. of the 11th International Conference on Parallel and Distributed Computing, Applications and Technologies*, Wuhan, China, pp.249-254, 2010.

[15] S. Rubasinghe and A. Anandagoda, *WSO2 Data Services White Paper*, 2008.

[16] A. T. Manes, *SOA Principles Apply to Data Access and Management*, http://searchsoa.techtarget.com/news/article/0,289142,sid26_gci1266439,00.html, 2007.

[17] S. Gannouni, H. Mathkour and M. Beraka, Comparison criteria for data sharing approaches, *Proc. of the 6th International Conference on Computer Sciences and Convergence Information Technology*, Rep. of South Korea, 2011.

[18] A. Omicini and E. Denti, From tuple spaces to tuple centres, *Science of Computer Programming*, vol.41, no.3, pp.277-294, 2001.

[19] Dr.-Ing and K. Herrmann, *Asynchronous Middleware Tuple Spaces*, Distributed Systems Department, Institute of Parallel and Distributed Systems, University of Stuttgart, 2009.

[20] M. H. Needleman, RDF: The resource description framework, *Serials Review*, vol.27, no.1, pp.58-61, 2001.

[21] S. Zhou, Exposing relational database as RDF, *Proc. of the 2nd International Conf. on Industrial and Information Systems*, Dalian, China, vol.2, pp.237-240, 2010.