A Critical Comparison for Data Sharing Approaches

Sofien Gannouni, Mutaz Beraka, and Hassan Mathkour
Department of Computer Science, King Saud University, KSA

Abstract: *Integrating and accessing data stored in autonomous, distributed and heterogeneous data sources have been recognized as of a great importance to small and huge-scale businesses. Enhancing the accessibility and the reusability of these data entail the development of new approaches for data sharing. These approaches should satisfy a minimal set of criteria in order to support the development of effective and comprehensive data sharing applications. In this paper, we first outline the four data sharing approaches and define a set of fundamental criteria for data sharing approach. Moreover, we investigate the motivation and importance of these criteria, and the inter-dependencies among them. Additionally, we compare the existing data sharing approaches based on the available options for each criterion.*

Keywords: *Data sharing approaches, data access, data integration, data consistency, performance, criteria options.*

Received January 2, 2012; accepted September 23, 2012; published online January 29, 2013

1. Introduction

The variety of data sources and vast amount of data hosted by these sources mean that they play a very important role in small and large-scale business [16], as well as in research. Sharing of heterogeneous, autonomous, and remote data sources has long been a hot topic, and has added value to both personal users and companies. This type of sharing allows us to take advantage of enterprise data stored in these sources, and opens up opportunities to integrate data from multiple data sources to gain a holistic understanding of data integration [4]. Thus, the demand for sharing existing data sources is more important now than ever before.

Due to the different data sharing approaches reviewed and explained in [9], we need a set of criteria that must be considered and taken into account before choosing one of these approaches to develop a new data sharing application. Moreover, data source sharing, or simply data sharing, requires at least support for a set of fundamental criteria to meet the minimal requirements. So far, such criteria have not been discussed in current research. They haven't been described or explained in terms of supporting data sharing or for examining data sharing approaches that have been proposed during the last decade. In this paper, we present four fundamental criteria that must be satisfied when selecting one of the data sharing approaches for use in developing and implementing data sharing applications.

Different data sharing approaches have been introduced and applied in different computing environments. These approaches vary in terms of concepts and standards used in implementations thereof. The data sharing approaches under consideration are: transaction processing monitor, tuplespace, resource description framework, and data service approach [9]. Transaction Processing Monitor (TPM) is a database middleware based on transactions that allows users to submit their queries, which are then processed and executed over multiple database servers as transactions [9]. Tuplespace is a coordination language that provides shared memory space into which users can post data, or from which data can be retrieved [9]. Resource Description Framework (RDF) is a W3C standard for describing, representing and storing data from heterogeneous data sources as web resources on the web [9]. Data Service Approach (DSA) relies on a Service Oriented Architecture (SOA) to expose heterogeneous data sources as data services [9]. A preliminary study of these data sharing approaches with the fundamental criteria has been presented in [10]. However, [10] does not specify for none of them neither options nor mechanisms.

It is not obvious what the basis and the criteria upon which a specific approach would be more appropriate for a given application. It is desired, therefore, and beneficial to determine a set of criteria that can guide users to adopt the suitable data sharing approach for the application at hand. To the best for knowledge, no attempt has been in this regard. In an attempt to determine an applicable set of comparative criteria, we discuss in this paper four evaluation criteria namely, data access, data integration, data consistency and performance, and show how they can be used to adopt a given data sharing approach for the application at hand. We also shed light on the inter-dependencies among them. Additionally, we compare the existing data sharing approaches based on the available options for each criterion.

In the following sections, we describe four evaluation criteria that should be supported by different data sharing approaches and compare between these approaches based on the available options for each criterion. We strongly believe that any approach to data sharing should aim to satisfy these criteria, albeit without limiting the minimal criteria to only these. These criteria help developers to decide, after a review of the data sharing approaches, which approach is the most suitable to be applied during the development of a new data sharing application.

2. Data Access

The most common demand from enterprises and applications is the access to enterprise data stored in distinct heterogeneous data sources. These sources store a vast amount of data, which is either structured, such as a relational database, semi-structured such as an XML file, or unstructured such as a document [2, 3]. Additionally, the current demand for data sharing is more important than ever before. Access to data stored in heterogeneous data sources is needed to allow users to benefit from such data and to create opportunities for novel use of the data.

Providing access to many kinds of heterogeneous data sources effectively requires dealing with different formats, structures, and platforms, and the sources being distributed over a network. Additionally, functions for manipulating and accessing data stored in these sources should also be provided, so that users do not need to know the data formats, platforms, locations, distribution, and so on. Thus, providing a uniform solution for accessing and manipulating data stored in various data sources is not a trivial task. Instead, this challenging task should deal with all the issues noted above to provide transparent access to enterprise data stored in heterogeneous data sources. This transparency should also be extended to developers to enable them to provide this type of access [13].

Therefore, the solution should mask the heterogeneity between data sources and address some related issues such as custom-code, performance, and so on. A solution is to provide an abstract layer, preferably as middleware, which can access and manipulate data stored in various types of data sources in a generic way. This layer typically uses a variety of common interfaces to access different data sources such as Java DataBase Connectivity (JDBC) or Open DataBase Connectivity (ODBC), etc. [5].

Access to homogenous data sources is much easier than to heterogeneous data sources. In homogenous data sources, there is only one type of data format that must be dealt with, albeit with multiple vendors of the data source. For example, for structured data stored as a relational database, the vendors include Oracle, MySQL, etc. On the other hand, for heterogeneous data sources, different types of data formats as well as different types of data sources must be considered. In addition, accessing structured data stored in data sources is much easier than semi-structured and unstructured data. The representation of structured data is based on a well-defined schema such as data stored in database, whereas unstructured data differ both in the format and method of representation.

Finally, although providing access to heterogeneous data sources is very important, it doesn't absolve the need for data integration. This allows data to be moved at runtime as opposed to in batches, thereby enabling users to retrieve data from multiple data sources to gain a holistic understanding of data integration [19].

3. Data Integration

Data integration provides the ability for users to retrieve data from integrated data sources. The data extracted from these sources would not be possible if the data sources were viewed in isolation. Thus, providing a unified view of data integration helps us to avoid duplicating effort in gathering data and enables data to be retrieved from heterogeneous data sources that would otherwise be impossible [29].

Integrating data stored in autonomous, distributed and heterogeneous data sources is important for both business companies and personal users, as well as for researchers. It establishes a solid data foundation to meet strategic business intelligence and to achieve their objectives [17]. As for personal users, integrating shared data allows them to benefit from vast amount of data stored in data sources by querying and retrieving the data from multiple heterogeneous data sources.

Providing uniform access to multiple data sources is the main goal of data integration. This is significant in a variety of situations in both commercial and scientific research. Besides the main goal, data integration is followed by one of two sub goals. The first one applies when we know exactly the question we want answered and we need to know the amount of data available. The second goal understands the power of the data at large when the availability of data sources varies [4]. However, supporting data integration is not an easy task, and is typically accompanied by a set of challenges include exponential data growth, huge numbers of data sources and moves to cloud computing and high development costs [3, 6].

Most of the existing approaches, mechanisms, and techniques are based on a global schema or mediation schema to support data integration. They require great effort in dealing with different data formats, models, schemas, and query languages, as well as heterogeneity between data sources and their locations. Wang *et al.* [27], describe three methods to integrate data stored in heterogeneous data sources. Relational database integration method focuses on accessing relational databases and integrates data stored in these databases. XML data integration method provides a single uniform XML view of various data sources, and allows querying these sources using this view. Ontology data integration method provides a semantic integration infrastructure that uses ontology as the mediated schema for representing the semantics of the data sources. This mediated schema allows users to query data using a uniform query interface. There are also other data integration approaches, such as the data warehouse approach, that is, Extract, Transform, and Load (ETL), Enterprise Information Integration (EII), and so on [3].

Finally, integrating data using minimum efforts opens the door to share data that can satisfy most of the integration needs in current decade for both companies and researchers [20]. So, potentially the best solution to integrating data stored in heterogeneous data sources is to not rely on a global or mediation schema. Such a solution should provide virtual data integration based on set of proposed standards and technologies to integrate heterogeneous data sources at runtime.

4. Data Consistency

Data consistency means that data remain consistent, accurate and valid over time, and do not violate any application-logic constraints. This ensures that each user consuming the data sees a consistent view of the data, including visible changes made by user's own transactions and those of other users. However, the importance of data consistency comes from the fact that the users are retrieving and updating data that is stored on autonomous, distant and heterogeneous data sources; which means that users' statements are running simultaneously on several systems and may perform concurrent operations on the same data [22]. Hence, systems or applications may generate erroneous results or take inappropriate actions because they are handling inconsistent data [24]. The need for data consistency mechanisms has become very important to control transactions' processing and management among autonomous, distributed and heterogeneous data sources. Achieving data consistency means users can retrieve valid, accurate and up-to-date data at all times.

One of the earlier simple assumptions made by most application programmers was that to maintain data consistency, data should exist in one place and database transactions should be used as a method of ensuring the integrity of business actions [24]. This assumption simplified the code needed to manipulate data. However, the current demand for data consistency is more complex than this simple assumption, since heterogeneous data sources are typically distributed over a network. A uniform consistency model, the ACID model, which sets forth the four properties of Atomicity, Consistency, Isolation and Durability that all transactions must satisfy, is the best solution to ensure consistency of data in a distributed computing environment. The responsibility of the application programmer is to define transaction boundaries in a manner that is consistent with the application's behavior [24]. Accordingly, supporting this consistent model is not an easy task. Much effort is needed in implementing solid mechanisms that achieve these properties absolutely.

However, satisfying the ACID properties on homogeneous data sources is much easier than on heterogeneous ones. Moreover, achieving ACID properties on local data sources is easier than on distributed data sources. The difficulty of working in distributed environments stems from data sources being distributed over a network and the fact that the availability and reliability of these sources vary. Therefore, there are many things that need to be considered and taken into account when working in a distributed environment.

Finally, enterprise data stored in heterogeneous data sources are imported by unknown users and companies, and such data become part of the user or company's application for processing and/or consumption. For this reason, support for and satisfying ACID properties means that data remain consistent over time, resulting in high quality data, efficient data sharing, efficient data integration, findings based on solid evidence, and saving time and resources [8].

5. Performance

It is the property that measures amount of time required to accomplish the work based on the used resources [28]. In software engineering, it is considered as a fundamental property of a system under development and it is one of the non-functional requirements of that system [23]. Some analysts consider performance to be a functional requirement in certain systems.

In data sharing, performance is one of the most important criteria affecting the usage of the data sharing application. Users willing to share data stored in heterogeneous data sources need high performance for both integrating and accessing the data stored in these sources. Therefore, to provide a high performance data sharing application for accessing and integrating enterprise data, one first needs to understand the application's behavior, and then address the access and integration challenges to make it faster. Providing a single access layer that supports heterogeneous data sources and uses fast access interfaces, fast query execution, and concurrency control is an example of the data access challenges [3, 6]. If sources are reliant on slower interfaces, for example, this means a slower connection with the data sources and longer response-time for the given query. Furthermore, examples of the challenges of integrating data stored in data sources include the heavy reliance on global schemas, transformations, and mappings between different schemas. This heavy reliance on schema mappings, for example, tends to consume a lot of computer resources and provides a lower throughput.

Accordingly, meeting both current and future performance requirements is important for all commercial and open-source solutions for data sharing. A variety of mechanisms such as parallel processing and workload balancing can be used to accomplish the intended work effectively [17]. From our point of view, using parallel processing mechanisms in data sharing will provide a higher level of performance for accessing, retrieving and integrating data from multiple data sources. Working with heterogeneous data sources in parallel and using an asynchronous mode in some situations will reduce the time required to realize the intended work as well as allow complex data integration to be carried out. Additionally, using techniques that capture data changes, one can easily track and extract the changes that have occurred in the correct order to the relevant fields in the data source since the last extract [17].

Finally, everything affects performance: from the system itself to all underlying layers, such as the operating system, middleware, and so on [28]. In addition, performance affects the usability, reliability and availability of the system. In data sharing, performance also affects the access and the integration of data stored in heterogeneous data sources. Thus, providing a high-level of performance results in a high performance data sharing application that provides effective and efficient data sharing, as well as short response-time for given work, high throughput, low overhead, low resource usage and high reliability and availability of the system.

6. Inter-Dependencies between Evaluation Criteria of Data Sharing Approaches

Figure 1 shows inter-dependencies between criteria.

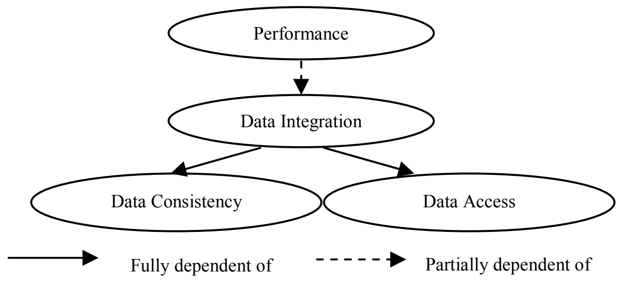


Figure 1. Inter-dependencies between criteria.

Data integration criterion is fully dependent of the data access criterion for the following reasons:

- Without providing the ability to access data stored in data sources, it is not possible to retrieve data for subsequent integration.
- Accessing heterogeneous data sources for data integration involves dealing with different data formats, schemas, and so on.

The data integration criterion is fully dependent on the data consistency, since integrating inconsistent data means poor integration and useless data. The performance criterion is partially dependent of the data integration criterion. Indeed, there are several options that could be adopted to implement the data integration criterion. The performance of each of these options depends also on the mechanisms and techniques, such as parallel processing techniques, that could be used to accomplish the option.

7. Options for Evaluation Criteria of Data Sharing Approaches

7.1. Data Access

It has two options as follows [3]:

- *Homogeneous:* This option supports one type of data sources from various vendors meaning the same data model but different implementations.
- *Heterogeneous:* This option supports different types of data sources from various vendors meaning different data models with various implementations.

7.2. Data Integration

It has three available options as follows [21, 27]:

- *Structured Data Integration Approach:* It aims to integrate heterogeneous structured databases. It doesn't capture semantic behind database schema; only reflecting database structure. So, integrating databases is obviously hard.
- *Semi-Structured Data Integration Approach:* XML as a standard language used to represent different kinds of data, could be used to support data integration. This approach provides a single uniform XML view of various data sources, and allows users using this view to submit their queries to retrieve data from these sources.
- *Ontology Data Integration Approach:* Concerning to semantic Web, this approach has been proposed to support data integration using ontology. It captures the semantic behind the data sources schema and provides a semantic integration infrastructure that uses an ontology as the mediated schema for representing semantics of data sources.

In addition, data integration criterion has two available mechanisms as follows [3]:

- *Extract, Transform and Load (ETL):* It is designed to process huge amount of data stored in heterogeneous data sources and in data warehousing. It extracts data from different daily operational heterogeneous data sources, transforms and loads it into a target data source.

- *Enterprise Information Integration (EII):* It is the process of data integration which provides a single virtual interface to uniform data access for heterogeneous data sources. From the viewpoint of data integration, EII supports data integration using the federated databases approach.
- *Data Integration through Service Composition:* It is the process of data integration, which enables the different data sources to publish services providing access to their related data. Users are willing to compose a set of services, publishing by the data sources, in one business process and execute this process to retrieve and integrate data from multiple data sources. Hence, the data integration is done through the service composition process unlike other techniques.

7.3. Data Consistency

It has two options models as follows [5, 24, 25, 26]:

- *ACID Transactions:* In this model, each transaction must satisfy the ACID properties in order to guaranty the database consistency and to guard the user's operations against hardware and software failures [5].
- *Eventual Consistency:* It is used in distributed shared memory, distributed transactions and optimistic replication [25, 26]. The system will eventually become consistent [24]. It means that given a sufficiently long period of time over which no changes are sent, all updates can be expected to propagate eventually through the system and all the replicas will be consistent.

ACID transactions model supports the following options models [1, 18]:

- *Flat Transaction Model:* It does not allow transactions to be nested within other transactions. Therefore, two transactions are executed in different scopes.
- *Nested Transaction Model:* In this model, a transaction is composed of an arbitrary number of sub-transactions that may be executed concurrently. Top-level transactions must satisfy ACID properties. The dependencies between the sub-transactions and the top-level transaction are fixed in the model.
- *X/Open Distributed Transaction Processing (DTP) Model:* It is an industrial standard defined by the open group consortium. The X/Open allows multiple applications to share resources provided by multiple resource managers and allows their work to be coordinated into global transactions [1, 15]. A two-phase commit protocol with presumed rollback is performed across all the involved resource managers to assure global atomicity
- *Flexible Transaction Model:* It is trying to overcome the restrictions of previous models It is suitable for structured and long-running transactions and it supports both ACID and non-ACID transactional requirements. Therefore, it gives the flexibility to specify which sub-transactions should satisfy ACID properties and which aren't. Therefore, the dependencies that exist between a top-level transaction and its sub-transactions are no-longer fixed in the transactional model; they are specified by user.
- *Business Transaction Model (BTP):* It is published by the OASIS Business Transaction Technical Committee (BTTC) as a standard for coordinating transactions between applications controlled by multiple autonomous parties. No single party controls all resources needed in a business transaction. Parties manage their own resources but coordinate in a defined manner to accomplish the work scoped by a transaction. Individual service providers either agree to join a transaction or not.

Eventual consistency model supports following options:

- *Immediate Refresh:* In this option, distributed replicas are updated automatically at the end of each transaction that updates the original data source.
- *Periodic Refresh:* In this option, replicas are periodically updated.
- *Differed Refresh:* In this option, replicas are updated when someone attempts to retrieve data from them.

7.4. Performance

Of course, we can easily know that it is difficult to determine general options that could be supported by data sharing approaches to achieve high performance for developing data sharing solution. Therefore, we will present performance metrics that help developers during the selection phase of an appropriate data sharing approach. These metrics are as follows [14]:

- *Overhead:* It is the additional processing required by the approach performs a regular task. This metric is sensitive to the following overhead sources:
  1. *Communication:* Its type between parties may form an overhead such as cost of initialization communication variables, shared communication data, memory allocation and so on.
  2. *Mode of Execution:* The cost of synchronous execution mode is different to those of the asynchronous mode [11].
  3. *Computation:* The way the computation is accomplished is an important indictor on the performance. It may be inherently sequential or parallel. Unlike the parallel processing mode, the sequential computation doesn't benefit from the possibility of multicore microprocessors.

- *Memory:* The amount of the allocated memory and its usage is very important indicator on the level of overall performance.
- *Contention for Resources:* It's a special case of overhead when consumers compete for shared resource. So, its effects can often lead to slowdown that the result is worse performance.

8. Comparison between Data Sharing Approaches

We compare hereby the different data sharing approaches according to the four evaluation criteria described above. The result of this comparison is summarized in Table 1.

- *Data Access*

  TPM and tuplespace were originally proposed to support one king of data source, which is a structured data source relational database. Whereas, RDF and DSA approaches were originally proposed to hide the heterogeneity between various data sources.

- *Data Integration*

  TPM uses structured data integration approach along with one of the two options ETL or EII. Therefore, it can support data integration through a mapping process of all extracted data and load it in one target database, or it can build a virtual data-integration layer. Tuplespace uses structured data integration approach with one option, which is ETL option to support data integration. It may not be able to support EII option. It allows users to post their data in a shared space; so building a mediated schema is impossible. RDF, as data sharing approach dedicated to the semantic Web, is adopting the ontology data-integration approach to support data integration. This approach allows the developer to adopt the EII option by specifying a mediated schema in order to support data integration. Moreover, developers of data sharing application may integrate data from different data sources through the service composition option.

  The DSA approach adopts the semi-structured data integration approach to support data integration. It relies on Web service technology as standard language that heavily relay on XML standard. It enables the adoption of EII option since the developer may define a mediated schema in order to support data integration. However, the service composition option is the natural process to integrate data from various heterogeneous data sources for the DSA approach. Unfortunately, RDF and DSA aren't supporting ETL option because they aim to benefit and reuse existing heterogeneous data sources and integrating them with minimal effort.

- *Data Consistency*

  Satisfying the data consistency requirements between homogeneous data sources is much easier than heterogeneous data sources. In addition, it is much easier to satisfy ACID properties in centralized environment rather than decentralized environment. However, TPM supports ACID transaction models such as flat transaction model, nested transactions model, flexible transactions model or X-Open transaction model. But, it doesn't support business transaction model, since there is no business processes.

  Tuplespace supports both flat model and X/Open ACID models. RDF supports ACID models such as nested transactions model, flexible transaction model or X/Open transaction model. But, it doesn't support flat model since data sources are distributed across the network. In addition, RDF supports eventual model with one of its options, because replicas are residing on different sites and the requirement of making the overall system consistent is very important.

  DSA support both models, ACID and eventual with their options. It supports well the business transaction model since it relies on SOA, which exposes data sources as services. So, it requires creating a business transaction that involves different business parities [24].

Table 1. Summary of comparison between data sharing approaches based on proposed evaluation criteria.

	TPM	**Tuplespace**	**RDF**	**DSA**
Data Access	Homogeneous (relational databases)	Homogeneous (relational databases)	Heterogeneous (from structured to unstructured data sources)	Heterogeneous (from structured to unstructured data sources)
Data Integration	Structured data integration approach	Structured data integration approach	Ontology data integration approach	Semi-structured data integration approach
	ETL or EII	ETL	EII or data integration through service composition	EII or data integration through service composition
Data Consistency	ACID model with one of the following models: Flat transaction model, nested transaction model, X/Open model or flexible transaction model	ACID model with one of the following models: Flat transaction model or X/Open model	ACID model with one of the following models: Nested transaction model, X/Open model or flexible transaction model or Eventual model with one of the following options: Immediate refresh, periodic refresh or differed refresh	ACID model with one of the following models: Flat transaction model, nested transaction model, X/Open model, flexible transaction model or business transaction model or Eventual model with one of the following options: Immediate refresh, periodic refresh or differed refresh
Performance	No options			

9. Related Issues for Data Sharing Approaches

In addition to the criteria explained above there are currently some issues that should be considered for the selection of a data sharing approach, and for the implementation of a new data sharing system. In this paper, we provide a summary of these important issues, which are the infrastructure environment, appropriate standard technologies, security and execution mode.

The main aim of data sharing is to share a huge amount of data distributed across network. Therefore, the important issue is either to support this type of sharing in a centralized environment or in a decentralized heterogeneous environment. The Peer-to-Peer (P2P) infrastructure has presented creditable advantages than centralized environments, avoiding both computational performance and information update bottlenecks, and providing other significant issues including scalability, reliability, availability and so on [12]. However, a valuable result for both business companies and users is to selecting an appropriate data sharing approach supporting P2P infrastructure efficiently, and developing a new data sharing system based on this approach in P2P environment to share heterogeneous and autonomous data sources across the network.

Another issue related to the development of a new data sharing solution is the choice of an appropriate standard technology that will be used to implement the main concepts related to data sharing approaches for. This technology may be proprietary, or an open standard technology. Moreover, the implementation of the technology should be selected carefully. However, selecting a standard technology should satisfy at least set of main characteristics, which are well-defined, enables reuse of IT existing assets, platform independent and provides a clear API for the most widely used programming languages.

Yet another important issue is security. The requirements of data sharing systems in terms of security are organized into the following areas: availability, authenticity, anonymity, and access control [7]. Supporting and implementing these techniques in order to apply and enforce the security policy on the developed system is directly affecting the performance.

Another issue that is valuable to enhance the overall system performance is the execution mode. There are two modes, which are synchronous and asynchronous. Synchronous mode is blocking until the result becomes available, whereas, asynchronous mode is not blocking, it fires call, continue working and handles result when it is available. However, supporting one or both modes depend on the selected approach and the system requirements. Nevertheless, in some cases, supporting parallel technique along with asynchronous mode has advantages for the overall system performance.

10. Conclusions

In this paper, we identified and explained four comparison criteria, data access, data integration, data consistency and performance. Each of these criteria may be accomplished in different ways. We have studied and discussed the available options for every criterion. This study allows us to compare the existing data sharing approaches based on the proposed criteria and their related options. This comparison can help the developer to select the most appropriate data-sharing approach for the development of a new comprehensive data sharing solution.

In the near future, we will present the open issues such as security issues, robustness, etc., that are related to data sharing approaches in order to provide a concrete study and a deeper analysis between these approaches. We expect that the current and the future study will become a powerful guide for researchers and developer to build, design and implement a comprehensive data sharing solution.

Acknowledgment

This work was supported by the Research Center, College of Computer and Information Sciences, King Saud University.

References

[1] Anna-Brith A., “The xTrans Transaction Model and FlexCP Commit Protocol,” *Technical Report*, University of Tromsoe, 2006.

[2] Bloomberg J. and Goodson J., “Best Practices for SOA: Building a Data Service Layer,” *SOA World Magazine*, vol. 8, no. 5, pp. 1-6, 2008.

[3] Bloomberg J. and Schmelzer R., “The Data Services Layer: Building a Solid Foundation for SOA,” available at: http://www.zapthink.com/ 2009/06/23/ video -the- data-services-layer building-a-solid-foundation-for-soa/, last visited 2009.

[4] Brazhnik O. and Jones J., “Anatomy of Data Integration,” *Journal of Biomedical Informatics*, vol. 40, no. 3, pp. 252-269, 2007.

[5] Cappellen M., Cordewiner W., and Innocenti C., “Data Aggregation, Heterogeneous Data Sources and Streaming Processing: How Can XQuery Help?,” *in Proceedings of the IEEE Computer Society Technical Committee on Data Engineering*, pp. 1-8, 2008.

[6] Chandrasekaran S. and Alvarez P., “2011: The Year of Data Virtualization,” available at: http://www.sfdama.org/Presentations/2011/2011 %20Year%20of%20Data%20Virtualization%20-

%20Best%20Practices_20110309_SFDAMA_Denodo.pdf, last visited 2011.

[7] Daswani N., Garcia-Molina H., and Yang B., "Open Problems in Data Sharing Peer-to-Peer Systems," *in Proceedings of the 9th International Conference on Database Theory*, Italy, pp. 1-15, 2003.

[8] Eynden V., Corti L., Woollard M., Bishop L., and Horton L., *Managing and Sharing Data*, UK Data Archive, University of Essex, Wivenhoe Park, 2011.

[9] Gannouni S., Mathkour H., and Beraka M., "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.

[10] Gannouni S., Mathkour H., and Beraka M., "Comparison Criteria for Data Sharing Approaches", *in Proceedings of the 6th International Conference on Computer Sciences and Convergence Information Technology*, Seogwipo, pp. 442-445, 2011.

[11] Goldsmith B., "Distributed Computing and Communication in Peer-to-Peer Networks," *PhD Thesis*, University of Tasmania, 2010.

[12] Haase P., Broekstra J., Ehrig M., Menken M., Mika P., Plechawski M., Pyszlak P., Schnizler B., Siebes R., Staab S., and Tempich C., "Bibster- a Semantics-Based Bibliographic Peer-to-Peer System," *in Proceedings of the 3rd International Semantic Web Conference*, Berlin, pp. 122-136, 2004.

[13] Huang F., "Heterogeneous Data Source Access in Web Applications," available at: http://ee85.yi.org/cisc832/cisc832paper.pdf, last visited 2000.

[14] Lin C. and Snyder L., *Principles of Parallel Programming*, Addison-Wesley, San Francesco, 2009.

[15] Maabreh K. and Al-Hamami A., "Implementing New Approach for Enhancing Performance and Throughput in A Distributed Database," *International Arab Journal of Information Technology*, vol. 10, no. 3, pp. 290-296, 2013.

[16] Manes T., "SOA Principles Apply to Data Access and Management," available at: http://searchsoa.techtarget.com/news/1266439/SOA-principles-apply-to-data-access-and-management, last visited 2007.

[17] MAS Strategies., "Data Integration: Creating a Trustworthy Data Foundation for Business Intelligence," *BusinessObjectsTM an SAP® Company*, White Paper, 2008.

[18] McGovern J., Tyagi S., Stevens M., and Mathew S., *Java Web Services Architecture*, Morgan Kaufmann, USA, 2003.

[19] MicroStrategy, "Accessing Heterogeneous Data Sources using MicroStrategy MultiSource Option," available at: http://www.ts.avnet.com/clientsolutions/accessing_heterogeneous_data_sources_using_microstrategy_multi-source_option, last visited 2011.

[20] Miller B., "Data Integration Demand will Grow in 2008," available at: http://www.zdnet.com/news/data - integration - demand - will - grow - in - 2008/181807, last visited 2011.

[21] Mostefai S., Bouras A., and Batouche M., "Data Integration in a PLM Perspective for Mechanical Products," *International Arab Journal of Information Technology*, vol. 2, no. 2, pp. 141-147, 2005.

[22] Oracle, "Oracle9i Database Concepts," available at: http://docs.oracle.com/cd/ B10501_01 / server.920/a96524.pdf, last visited 2002.

[23] Park D. and Kang S., "Design Phase Analysis of Software Performance using Aspect-Oriented Programming," *in Proceedings of the 5th Aspect-Oriented Modeling Workshop in Conjunction with UML Lisbon*, Portugal, pp. 1-7, 2004.

[24] Sholler D. and Schulte W., "Data Consistency and SOA: Old Challenges Rear Their Ugly Heads," available at: http://www.gartner.com/id=1183313, last visited 2009.

[25] Terry D., Theimer M., Petersen K., Demers A., Spreitzer M., and Hauser C., "Managing Update Conflicts in a Weakly Connected Replicated Storage System," *in Proceedings of the 15th ACM Symposium on Operating Systems Principles*, USA, pp. 172-182, 1995.

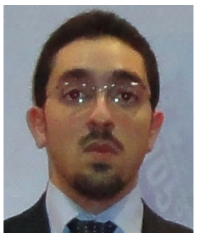
[26] Vogels W., "Eventually Consistent," *ACM Queue*, vol. 6, no. 6, 2008.

[27] Wang J., Lu J., Zhang Y., Miao Z., and Zhou B., "Integrating Heterogeneous Data Source using Ontology," *Journal of Software*, vol. 4, no. 8, pp. 843-850, 2009.

[28] Woodside M., Franks G., and Petriu D., "The Future of Software Performance Engineering," *in Proceedings of Future of Software Engineering*, USA, pp. 171-187, 2007.

[29] Zeng J., "Research and Practical Experiences in the Use of Multiple Data Sources for Enterprise Level Planning and Decision Making: A Literature Review," *Technical Report*, Center for Technology in Government University, Albany, 1999.

Sofien Gannouni received his Msc degree in computer science from Paul Sabatier University France, and his PhD degree in computer science from Pierre & Marie Curie University France. Currently, he is an assistant professor at College of Computer and Information Sciences, King Saud University. His main research interests include service-oriented computing, distributed computing, parallel processing, middleware grid computing and cloud computing.



Mutaz Beraka received his BSc and MSc degrees in computer science from University of Petra, Jordan and King Saud University, KSA respectively. Currently, he is a PhD student at College of Computer and Information Sciences, KSU. His main research interests include service-oriented computing, Web service technologies, cloud computing, distributed systems, intelligent systems and software engineering.

Hassan Mathkour received his MSc and PhD degree in computer science from the University of Iowa, USA. Currently, he is a professor and the vice dean for development and quality, College of Computer and Information Sciences, KSU. His main research interests include service-oriented computing, distributed computing, artificial intelligence, bioinformatics, image processing and software engineering.