Dynamic Data Services

Data access for collaborative networks in a multi-agent systems architecture

Ruhollah Farchtchi
Department of Computer Science
George Mason University
Fairfax, USA
rfarchtc@gmu.edu

Abstract— **Community networks require advanced domain-independent services that can be tailored for their use. Multi-agent architectures have been explored as an architectural approach to achieving a community network or system. Building on the work that was done with, Enterprise Knowledge Agents [6], this paper looks at using data services and the challenges with respect to enabling a platform for query and representation of agent observed knowledge in the enterprise. The EKA paper describes a notional architecture for a multi-agent based system that catalogues and coordinates knowledge and collaboration in the enterprise. In that paper case agents are defined to collect and organize information for an individual in the enterprise. The challenge is that the agents require a method of storing the data as well as allowing the agents to query each other to understand what data may exist with other individuals in the organization. This paper explores the use of dynamic data services in solving the challenges with an environment where data must be shared and where multiple representations of data may exist as in the EKA system.**

Data Services; Data Architecture; Knowledge Management; Collaboration; Service Oriented Architecture; Semantic Web; Data Provenance; Community Network; Interoperability and Integration

I. INTRODUCTION

A major component of Service Oriented Architectures (SOA) is that they model business services to allow for composition of business functions into enterprise applications. Organizations today are highly dependent on and driven by the data they produce and consume. Modeling of the data in an organization to allow for use and composition in support of a SOA is no trivial task. Organizations may be widely distributed and have varying data requirements within departments. It is in these scenarios that organizations need to service-enable data within the organization allowing for data compositions to be made. In this paper, I propose a data services architecture to provide data access methods and a set of finite operations that can be executed over the data. Data services allow for query and response through a mechanism of query expansion. This is accomplished through methods that allow for query of the schema as well as the data. Applications that utilize data services would not need explicit knowledge of the underlying schema to initiate a query, yet are still able to access additional entities and attributes as needed. As organizations adopt more varied types of data beyond structured relational data (e.g., XML, Text, Hierarchical, RDF, OWL), the need for data services that standardize access to the data is required.

A. *Scope of Research*

The goal of this research is to define the characteristics of data services and propose a notional architecture for data services. To limit scope, the focus of this paper is on the query and integration of knowledge bases in a multi-agent system such as the EKA system. The EKA system is used as an example application to demonstrate the use of data services in an application. The use of data services is not limited to the EKA system. Data services can be applied anywhere where a similar use case may exist, however, for the purposes of this paper, we focus on the interaction between agents in the EKA system and how data might be discovered, queried, and reused. To put this in the context of the EKA system, let's revisit the EKA system business case.

Sally is a knowledge worker within her organization. She serves as a subject matter expert on Uzbekistan. Steve is a knowledge worker in the same organization and conducting some new research on countries in Central Asia. Neither Sally, nor Steve knows about each other or the work they are doing. By using the system described below,

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8359
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8359

agents learn about what Sally and Steve are doing. Steve's agent is able to conduct a search based on what the agent knows about Steve's work. Agents for Sally are able to conduct a search based on what the agent knows about Sally's work. Agents for Sally broker collaboration agreements with agents for Steve and determine that it may be beneficial for Sally and Steve to work together. On Sally's knowledge portal she receives an invitation to view some of the work Steve is doing. On Steve's knowledge portal he receives an invitation to view some of the work Sally is doing. Steve decides to include Sally's work in his own task and collaborates with Sally. Sally's Agents are able to supply Steve with services to integrate her collected knowledgebase with the work he's doing. The system provides the tools for Steve to integrate that knowledge as he sees fit. Together Sally and Steve now are able to achieve a greater deal of collaboration and information sharing within their organization [6].

The EKA system used agents to model case information based on users' subject matter expertise. In doing so it was envisioned that agents would need to have the ability to query each other for information facilitating information sharing and collaboration. The EKA work did not go into the specifics of how data would be queried and integrated and made several assumptions on the structure of the data. In reality the agents would need to be able to abstract the specifics of several different types of data sources. The data sources could range from multiple RSS feeds with different elements to relational databases with complex schemas.

B. *Research Importance*

Heterogeneous data exists in several system environments and architectures. More often than not it is addressed by producing additional copies of the data integrated into a single schema. This has caused great difficulty in data integration and interoperability. The use of global schemas has proven to be an effective mitigation, but lacks scalability when the number of data sources grows. Data must be removed of semantic inconsistencies before presentation to the user, causing overhead on either query execution or when the data is gathered and integrated. Finding data in the environment is difficult unless you already know its location. Experienced knowledge workers are too often required to create and query these knowledge bases leaving effective knowledge discovery in the hands of too few. Ill-posed queries and inaccurate schemas or metadata often result in poor recall preventing discovery of unknown knowledge. Clients wishing to query the data must have knowledge of the global schema, causing increased point-to-point integration scenarios. This results in a decreased ability to discover additional relevant attributes to a query at runtime. These attributes must be discovered and integrated at design and integration time.

II. Problem Context and Related Research

A. *Service Oriented Architecture*

How do data services fit in a service-oriented architecture?

There are many aspects of a SOA that provide benefit to enterprises and businesses. One such benefit in a SOA implementation is the use of Web services as an integration tool to allow applications or services implemented using multiple different platforms to work together. Web services in SOA environments make use of the Simple Object Access Protocol (SOAP) to communicate. The SOAP protocol defines an XML-based format for messages to be passed in Web service invocation. Various application areas use WS from B2B communication, enterprise application integration, e-Science all make use of Web services whether they are utilized in a SOA or not. Many of these WS scenarios, at some point, have a need to transfer large amounts of data. Embedding such data sets in SOAP messages is possible, but not a suitable solution from a performance perspective [4]. Habich, Richly, and Grasselt [4] describe the performance impact data transfer takes when using SOAP messages to transmit the data. They find that when the data and the SOAP message structure increase in complexity, the performance of the SOAP transmission gradually gets worse. Performance is also impacted with the process of XML serialization and deserialization. Serialization (and deserialization) presents a bottleneck when processing large messages. This is mainly due to the fact that the XML processing is a time and main-memory consuming process [4]. Habich et al., describe a framework for data services that utilize web services to provide an abstract interface to data. Their framework consists of a data

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8359
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8359

access layer, a data mediation service, and a data controller. The data controller serves as a register for different data schemas. The controller assists with creation of the data objects in the data access layer based on the registered schema. Web service calls are broken out into two components as messages for data and messages for parameters [4]. Service invocation involves specifying the definition for input/output of the data as parameters based on available schema information. Habich et al., [4] focus on providing an efficient method for moving the data using a Web services framework. They effectively push the data transfer off to other mechanisms to avoid the impact of the XML serialization and deserialization by allowing the encapsulation of other data movement tools, such as ETL tools, by their framework. This does lead to efficiencies in the movement of the data, but does not account for clients that may not have knowledge of the data they will receive. It is assumed that the client understands the underlying schema. Additionally, if data is queried and retrieved repeatedly, it is unclear if any efficiencies can be gained by incrementally updating the data already with the client. To realize these efficiencies with a data service, it is essential to have a record of where the data originated, how it was manipulated, and where the destination for the data is. This concept is more commonly referred to as data provenance.

Fujun Zhu et al., suggest and propose a service-oriented data integration architecture to provide a dynamically unified view of data on demand from various autonomous, heterogeneous, and distributed data sources [5]. They propose data access services that can be semantically described and discovered. In this paradigm, service providers publish their data sources as data access services, which may then be discovered, bound at the time they are needed, and disengaged after use [5]. Their aim is to provide a service that virtualizes different data sources. This service can be instantiated several times binding data at instantiation. Each instance can represent a different set of data depending on how the request for service was provided.

The authors have built a number of prototypes in support of the Integration Broker for Heterogeneous Information Sources (IBHIS) project. Their work has shown that WS provide a good infrastructure layer for data services, but that a higher-level broker is necessary to be effective [5]. The main argument for this is that WS infrastructure does not allow for dynamic discovery of services and late binding of data sources. In support of this argument they state that UDDI registries are designed primarily for human use rather than for a computer. Dynamic service discovery requires something other than UDDI due to lack of support for semantics [5].

B. *Data Structure*

How should the data for data services be structured?

Services in an enterprise require access to data in multiple different structures and formats. It is unlikely that a system will deal with data in a single format. This is especially true as enterprises move to a SOA and try to leverage existing data stores rather than produce additional data silos. Semantics and data have gained increased importance as data is leveraged across services. According to McClean et al., [2] the integration of semantically heterogeneous data has become increasingly important due to the rise of web services and the Semantic Web. Organizations use different classification schemes to describe their data. The data is obviously related as in the case of the Organization for Economic Co-operation and Development (OECD) and the Nomenclature for the Analysis and comparison of Budgets and Scientific Programmes (NABS) [2], but there are differences in the details of the data across several dimensions (e.g., granularity). Organizations like Eurostat need to be able to query these data sources on aggregate to produce new information. McClean et al., [2] describe an approach to integration using local ontologies that map to a common ontology integrating the data sets.

Hose et al., [3] propose the use of Peer Data Management Systems (PDMS) for loosely coupled and large-scale data integration scenarios. Their work lays out the current work in the PDMS area and lends itself to providing a basis for several considerations with database structure for data services. In their work, they describe system model characteristics [3] that would allow for a PDMS to function and compare and contrast several methods in research literature. They address attributes like data model, topology, mapping language and query language. Similarly, in a SOA the same attributes would need to be addressed to allow for data services.

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8359
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8359

Data provenance also has an impact on data structure, as there are two primary methods of structuring data provenance information [19]. The provenance of data can be described as an annotation through a rich set of metadata. On the opposite end of the spectrum, the provenance of data can be described using inversion [19], or through the functions that occurred to arrive at the data that is observed. Using annotative metadata is the more robust and descriptively rich way to represent provenance, although it may result in a larger data set than the data itself [19]. Conversely, an inversion method of representing data provenance is more compact, however, not all data transformations lend themselves to inversion. A defined function on a piece of data, for example, must be deterministic in order for the original piece of data to be derived from the results.

Due to a lack of standards for data provenance [22], the types and level of detail for data provenance is left to individual implementations. The data services would need to adopt an open standard for provenance and enable data sources to describe the provenance of the data robustly regardless of the approach i.e., annotations vs. inversion.

C. *Schema Query*

How do I query the schema and how does the service help me find what I'm looking for?

Hose et al., cover attributes of schema query as attributes of query planning and execution. Query planning attributes can be relaxation of completeness, pruning, or adaptive planning. Adaptive planning considers that it may be worthwhile to generate a query plan after polling PDMS peers [3]. In the data services paradigm, this would translate to querying the schemas of other services to determine the correct plan for the query. While it is not the purpose of this paper to cover query planning of data services, the method of adaptive query planning lends itself well to a loosely coupled environment like a SOA.

D. *Query Refinement*

Assumption: I'll never fully understand the schema I am trying to query.

How will the service help me identify what I'm looking for?

Hose et al., refer to scientific data grids where individual centers of research maintain their own data yet have common schema attributes that unify them [3]. The ability to query based on common schema attributes can be augmented with the specialized attributes of the individual source of data. The problem lies in being able to take the query and understand the specialization that is there. In the case of data grids, an approach may be to make suggestions to the user for user based refinement. In this case a query on common attributes is returned as well as suggestions based on the specializations of the individual data providers. Depending on the user's choice, the query can then be refined by those additional criteria.

III. DATA SERVICES

To truly achieve a vision of a community network, it is imperative for both services and data to be available for reuse. Therefore, data services are essential to allow data to be reused and composed into new data and information products regardless of the applications that may implement other services that utilize the data.

A. *Characteristics*

In order to accomplish the vision of data services, some constraints must be placed on a data service implementation. These constraints represent the minimal characteristics required to allow for data to be discoverable, searchable, managed and linked. The table below describes the basic architectural constraints for data services.

TABLE I. DATA SERVICE REQUIREMENTS

Constraint	Description
Be service based.	Data services must fit into a Service Oriented Architecture allowing for reuse via service composition.
Allow for multiple schemas.	Data services must allow for multiple schemas to exist in the enterprise. While each service may represent a single schema, a single data service may allow for methods across multiple schemas.
Enable discovery and mediation.	Data services must allow for discovery of schemas. It is assumed that other services would not have prior knowledge of the schema implemented by the data service. Therefore the data service must allow for discovery and acquisition of schema elements by the client application. This includes having a possible hierarchy of domains, ontologies or schemas as well as services to broker and mediate exchanges. Differences in the semantics between schemas must be mediated.
Promote data hiding.	For data services to be effective in multiple scenarios it must not matter where the data lives or what the source was. The service abstracts the details of the underlying data.

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8359
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8359

Constraint	Description
Support schema evolution.	Data services should not be restricted to all or part of a governed static global schema. As the data services evolve they should evolve the global schema.
Preserve data lineage.	In a community network with reuse and composition of data in to new data products, the DNA or provenance of the data becomes increasingly important. Data services must provide a mechanism to provide the lineage of the data in the results they provide. Therefore the data service must have a standard mechanism to annotate data elements at multiple levels of granularity.
Support dissemination of the data	Data services not only need to provide query and response, but also need to be able to support a publish-and-subscribe model for dissemination of data, information, and knowledge products.

An abstract description of a data service is represented in figure 1. It describes the interfaces that must exist to address the architectural constraints for data services outlined in Table 1. Data services include five interfaces; three bi-directional interfaces and two unidirectional interfaces.

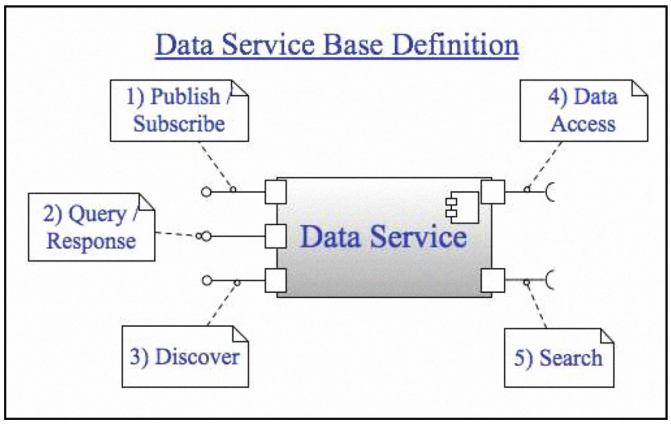

Figure 1. Data Service Base Definition

The data service interfaces are described as follows:

1. Publish/Subscribe - allows for data subscriptions given provided query criteria
2. Query/Response - provides query and results given a set of criteria
3. Discover - broadcasts service description on instantiation and responds to search requests
4. Data Access - connects to data sources based on either brokered exchange (e.g., other data service) or hardwired connection (e.g., ODBC to DBMS)
5. Search - Issues search requests to discover interfaces of data services

B. *Approach*

The Data Service described above needs to adhere to the architectural constraints of data services outlined in Table 1. Table 2 below, describes how our defined data service addresses the architectural constraints.

TABLE II. DATA SERVICES APPROACH

Constraint	Approach in Data Services
Be service based.	The data service exposes its methods via web services allowing multiple applications to utilize its methods and functions over the data. The interfaces support a service-oriented approach that allows for reuse via service orchestration and composition.
Allow for multiple schemas.	The data service has its own schema that it exposes and describes via RDFS or OWL. Consumer services that wish to use the data maintained by the data service can gain knowledge of the schema elements by obtaining the schema definition invocation of the discover interface. Exposure of the schema happens though the Query/Response interface. Here, services can query the schema and obtain relevant information.
Enable discovery and mediation.	Discovery is handled via a discovery interface that broadcasts the existence of the data service and listens for broadcast data requests. Data services promote schema discovery and mediation by recommending schema elements based on the broadcast search requests.
Promote data hiding.	The data service abstracts the data source from the application allowing for multiple types of data sources to be used. The data service can utilize other data sources as long as they are mapped to the data service's schema and described using RDF or OWL. This is an essential differentiator. A data service does not need to store the data itself. It is pointed to the data and then performs its methods over that data. This approach lends itself well to the Semantic Grid Architecture [23, 24] as it allows for operations over data with out data movement. The prerequisite being that the data sources are part of the grid fabric.
Support schema evolution.	The data service does not have any knowledge of a global schema. In some environments it may be beneficial for an application to have a schema specific to that application's needs. However, when representing data as a service, the data may be used by multiple applications and it is the consumer applications themselves that would map to schema elements.
Preserve data lineage.	Data services annotate the data sources they consume with metadata about the data sources. In addition, by using an RDF triple representation of data annotations are able to pass from data source to data source. This is an important aspect, as you will see with composition of data services. A data service is able to provide information on where the data came from though invocation of the query / response interface. Though this interface a service is able to identify what lineage data exists and can then query it.
Support dissemination of	The publish/subscribe interface supports data dissemination requirements of data services. Once

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8359
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8359

Constraint	Approach in Data Services
the data	a data service has been discovered and queried by a consumer service, the consumer service can choose to subscribe to that query criteria. Once subscribed a consumer service will received refreshed data as it meets the query criteria.

The data services approach allows for a great deal of flexibility and reuse of data as it is consumed, transformed, and exposed throughout a community network.

1) *Service Composition*

To further illustrate the concept of data as a service and service composition we define the interaction between data services and the client services that consume their output. We describe consumption of data services through the following wire diagrams. We first take simple consumption of a data service by another service. The nature of the consumer service is not important here; only the interaction between the interfaces is considered. For a consumer of data services to locate or discover an appropriate data service it must implement the search interface common to all data services. This will allow it to invoke the discover interfaces from available data services. This data service does not implement the search interface itself, as it is "hardwired" to a data source; in this case a RDBMS.

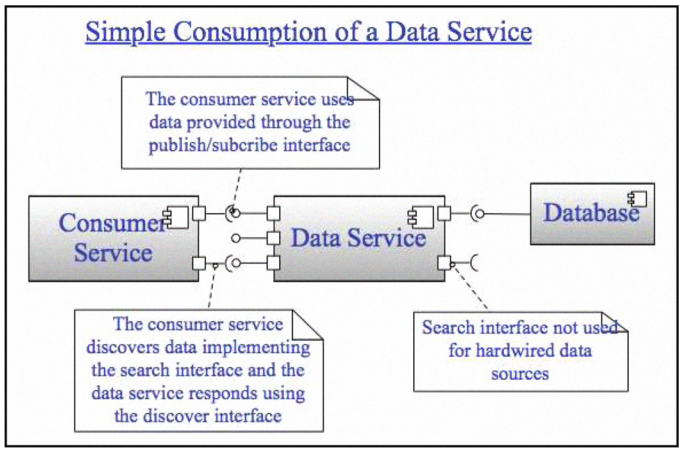

Figure 2. Data Service Consumption

As shown in figure 2, data services can be consumed in a very simple manner that abstracts the data source. This abstraction becomes more important as data services are composed together to create new data products and form composite data services. Figure 3 below describes several service composition scenarios. As data services can connect to and support operations over multiple data sources, they can consume other types of data sources. Working backwards in Figure 3 from right to left, we see that there are two data sources, a database and a data stream. The data stream represents data in motion while the database represents data at rest. Data Service 5 provides a representation of the data. Data Service 2 integrates Data Service 5 with its own database (Database 2) and, in turn, provides data to Data Service 1. Data Service 1 integrates data from Data Services 2, 3, 4 and from Database 1.

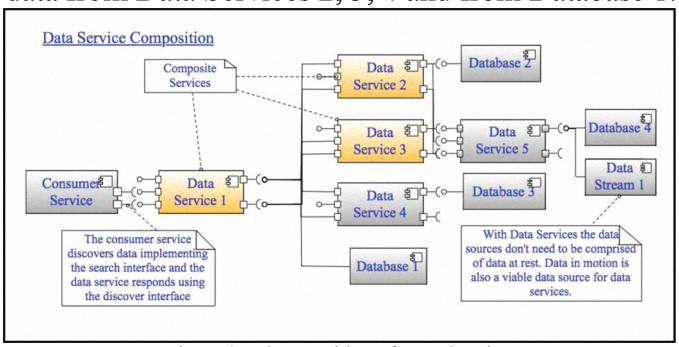

Figure 3. Composition of Data Services

Composition of services together allows for a great deal of reuse. As you can see from Figure 3, data services can be discovered, consumed, and used over and over again. The performance aspects of data service composition are not trivial. As data services are composed into other services and become more abstract from the data sources, performance can be problematic. It is the hypothesis of the author that, as data is exposed through data services, services will be supported by programming models and architectures that allow for massive parallelization such as MapReduce [1].

The majority of integration between data services is performed by the Query / Response interface. The Query / Response interface and the interactions with consuming services are critical to the use of data services.

2) *The Query / Response Interface*

Data services support two methods of query: 1) Simple Single-Query (SSQ) and 2) Advanced Query (AQ). The use case diagram in Figure 6 details the interaction between a case agent and the query / response interface of the case data service from the EKA system. Using the SSQ in the case data service, the case agent is able to issue a free-form query such as, "Senator Clinton" and an indicator of what type of result is necessary. The result could be a schema definition or it could be a result set of data meeting the query terms. Should the indicator

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8359
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8359

request a result of data matching the search terms the case data service does a search over the provided data set and returns the results. When the indicator specifies a schema result, the schema recommender is invoked and provides recommended schema elements as the result set. The recommendations are based on expanding the query to determine what schema elements are applicable. In he case of the query for "Senator Clinton", a simple expansion with an information extractor such as ClearForest [18] allows the recommender to provide schema elements that cover entities. Traversing the schema to identify adjacent nodes provides other data elements that are included in the results.

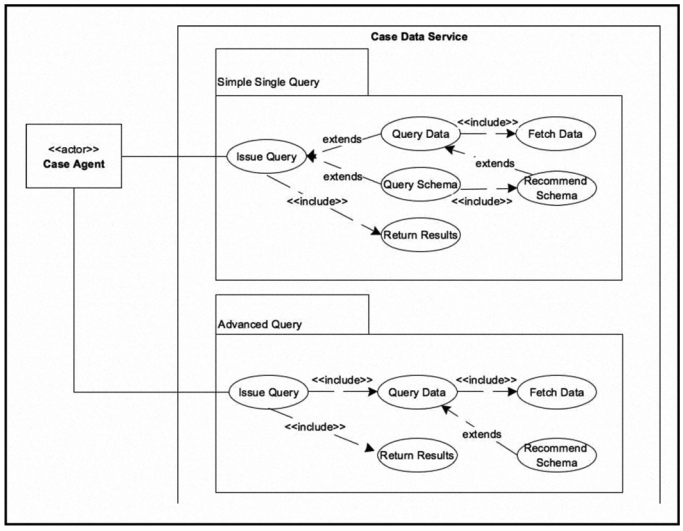

Figure 4. Use Case Diagram for Case Data Service

The AQ provides an advanced query utilizing the schema. In this use case the type of query is more structured using a SPARQL query. As the data services communicate schema definitions using RDF, they receive queries using SPARQL. This allows for a great deal of flexibility with queries provided. In this case a search for "Senator Clinton" may be represented in SPARQL as below:

```
PREFIX case:
<http://eka.com/caseOntology#Person
>
SELECT ?firstname ?lastname
WHERE {
  ?x case:firstname ?firstname ;
  ?y case:lastname ?lastname ;
     case:lastname = "CLINTON".
}
```

The above query is notional. It tests for the value LastName = "CLINTON" from the Persons entity in the case ontology. This would be determined from the schema definition provided by the case data service. When issuing an SSQ, the case data service would provide the URI's for the elements that would be used in a structured query of the case data.

IV. Future Research and Conclusion

This paper introduced several characteristics for data services. It provided a notional framework from which data services could exist. It also provided a structure of interaction between the data services and client services to query and retrieve results. This resulted in a deeper understanding of the necessary interactions and methods required to service enable data with out the use of global schemas and hardwired mappings.

Future work in this area would be to create a prototype working-version of data services to test the use cases described, and evaluate the methods that would be exposed. In addition, work on the schema recommendation service would be key. Exploring the relationships and the methods that used to accurately recommend schema elements for further query based on expansion of a simple free-form text query has multiple areas of applicability. The information retrieval arena is quite active in the area of query expansion from that approach.

Acknowledgment

Development of the concepts and thoughts in this paper would not have been possible with out the mentoring of Professor Larry Kerschberg of George Mason University.

References

[1] D. Jeffrey and G. Sanjay, "MapReduce: simplified data processing on large clusters," Commun. ACM, vol. 51, 2008, pp. 107-113.

[2] S. McClean, B. Scotney, P. Morrow, and K. Greer, "Integrating semantically heterogeneous aggregate views of distributed databases," Distrib. Parallel Databases, vol. 24, 2008, pp. 73-94.

[3] K. Hose, A. Roth, A. Zeitz, K. Sattler, and F. Naumann, "A research agenda for query processing in large-scale peer data management systems," Information Systems, vol. 33, 2008, pp. 597-610.

[4] D. Habich, S. Richly, and M. Grasselt, "Data-Grey-BoxWeb Services in Data-Centric Environments," Web Services, 2007. ICWS 2007. IEEE International Conference on, 2007, pp. 976-983.

[5] Fujun Zhu, M. Turner, I. Kotsiopoulos, K. Bennett, M. Russell, D. Budgena, P. Breretona, J. Keane, P. Layzell, M. Rigby, and Jie Xu, "Dynamic data integration using Web services," Web Services, 2004. Proceedings. IEEE International Conference on, 2004, pp. 262-269.

[6] J. Church and R. Farchtchi, "Enterprise Knowledge Agents: Agent-based Discovery of Like-minded Individuals," 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8359
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8359

Computing, E-Commerce and E-Services, Crystal City, VA: IEEE Computer Society, 2008.

[7] C. Carpineto, R.D. Mori, G. Romano, and B. Bigi, "An information-theoretic approach to automatic query expansion," ACM Trans. Inf. Syst., vol. 19, 2001, pp. 1-27.

[8] Xiaoling Wang, Aoying Zhou, and Guimei Liu, "Efficient Service Data Management in Dynamic Environment," Web Services, 2007. ICWS 2007. IEEE International Conference on, 2007, pp. 1040-1047.

[9] R. Chinta, S. Padmanabhuni, and K. Kunti, "Empowering next generation flexible operational data stores with service orientation," Next Generation Web Services Practices, 2005. NWeSP 2005. International Conference on, 2005, p. 6 pp.

[10] Sriram Anand, "Implementing Shared Data Services (SDS): A Proposed Approach," Services Computing, 2006. SCC '06. IEEE International Conference on, 2006, pp. 365-372.

[11] J. Hai, T. Yongcai, W. Song, and S. Xuanhua, "Scalable DHT-based information service for large-scale grids," 2008.

[12] P. Cudre-Mauroux, S. Agarwal, A. Budura, P. Haghani, and K. Aberer, "Self-organizing schema mappings in the GridVine peer data management system," Proceedings of the 33rd international conference on Very large data bases, Vienna, Austria: VLDB Endowment, 2007, pp. 1334-1337.

[13] W. Zhou, D. Jiang, P. Guillaume, C. Chi-Hung, and S. Maarten van, "Service-oriented data denormalization for scalable web applications," 2008.

[14] V. Niranjan, Sriram Anand, and Krishnendu Kunti, "Shared data services: an architectural approach," Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on, 2005, p. 690.

[15] J. van Hoof, "SOA and EDA: How to mediate semantics in an EDA."

[16] M. Antonioletti, A. Krause, N. Paton, S. Laws, J. Melton, and D. Pearson, "The WS-DAI family of specifications for web service data access and integration," ACM SIGMOD Record, vol. 35, 2006, pp. 48-55.

[17] S.T. Dumais, G.W. Furnas, T.K. Landauer, S. Deerwester, and R. Harshman, "Using latent semantic analysis to improve access to textual information," 1988.

[18] "Technology for Text Analytics." http://www.clearforest.com/Technology/TechnologyOverview.asp

[19] Y. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance techniques," 2006.

[20] W. Tan and U. Santa Cruz, "Provenance in databases: Past, current, and future," IEEE Computer Society Bulletin of the Technical Community on Data Engineering, vol. 32, Dec. 2007, pp. 3-12.

[21] P. Buneman, S. Khanna, and W. Tan, "Why and where: A characterization of data provenance," Lecture notes in computer science, vol. 1973, 2000, pp. 316-330.

[22] J. Myers, C. Pancerella, C. Lansing, K. Schuchardt, B. Didier, N. Ashish, and C. Goble, "Multi-scale science: supporting emerging practice with semantically derived provenance," ISWC 2003 Workshop: Semantic Web Technologies for Searching and Retrieving Scientific Data, 2003.

[23] P. Alper, O. Corcho, I. Kotsiopoulos *et al.*, "S-OGSA as a Reference Architecture for OntoGrid and for the Semantic Grid," in The 3rd GGF Semantic Grid Workshop, 2006.

[24] O. Corcho, P. Alper, I. Kotsiopoulos *et al.*, "An overview of S-OGSA: A Reference Semantic Grid Architecture," *Journal of Web Semantics*, pp. 102-115, 2006.

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8359
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8359