Efficient Domain-Specific Information Integration in Nexus

Thomas Schwarz[1], Nicola Hönle[1], Matthias Grossmann[1], Daniela Nicklas[1], Bernhard Mitschang
University of Stuttgart, Institute of Parallel and Distributed Systems (IPVS)
Universitätsstr. 38, 70569 Stuttgart, Germany
<firstname>.<lastname>@informatik.uni-stuttgart.de

Abstract

In this paper, we present the Nexus approach to efficient domain-specific integration of many loosely coupled data sources. A so called *information maximizing mediation middleware* (IMMM) has to cope with large data volumes and many queries, and at the same time achieve a tight semantic integration for the data instances. For efficiency and practicability reasons, we propose to use an extensible global schema and a limited domain-specific query language. This facilitates employing domain-specific semantic knowledge in the middleware: detect duplicates, merge multiple representations, aggregate and generalize information. Finally, we present a working prototype tailored to the domain of context-aware applications.

1. Introduction

The World Wide Web is a source for information that is structured for presentation to human readers and is thus mostly inaccessible to machines. The Semantic Web Initiative strives for a semantic description of this information. In this vision we soon will have not only a web of pages but also a web of objects. This vision addresses the automatic processing of information on the web, but not the problem that the information is scattered across numerous servers; locating and processing the right information is still a tedious task. Typically, software agents perform this task and integrate the information. Still, the application has to know about location and different data schemas of each provider to access.

In our approach, we go one step further: we introduce a transparency tier that hides the distribution of information providers from the applications. Now, applications perceive the impression of a single data source. They can benefit from new data providers without needing to change, and they are less affected by disappearing data providers.

Currently, we do not see the technology to do this in a generic way. However, our approach to achieve this goal is to create a mediation middleware, tailored to a single application domain (e.g. bio science, e-business, context-aware applications). With carefully chosen restrictions (global schema, limited query language) it becomes feasible to create a common data schema (upper ontology [8]). A more powerful middleware can be built that utilizes domain-specific semantic knowledge for advanced functionality and semantic information integration: detection of duplicate objects, merging of multiple representations, generalization or aggregation of data objects.

Applications from one domain typically utilize similar data models and their agents perform similar functions. Thus, in a common middleware this consensus can be exploited, so that applications need not provide their agents, but let the middleware perform this common functionality in a more convenient and efficient way.

After discussing related work in Section 2, we focus on requisites for building an information maximization mediation middleware (IMMM) and propose a general architecture for such a middleware (Section 3). We present our implementation of this general concept for the domain of context-aware applications in Section 4. In Section 5 we conclude the paper by highlighting the benefits of our design decisions and by giving directions of future work.

2. Related work

Information integration systems have been a research topic for many years. Today, even commercial products are available. In this section, we give an overview on existing products and approaches and classify them based on the following criteria:

- Dynamic: Is the system suited to cope with frequent addition and removal of data sources?
- Schema: What kind of schema does the approach offer to applications? This mainly determines the expressiveness of queries and the extensibility.
- Embedding costs: Effort to add a new data source.
- Integration mode: Kind of homogeneity that is achieved.
- Number of data sources: Magnitude of the number of data sources for typical applications.
- Result size: Magnitude of the maximum size of result sets for typical applications. Depending on the schema, a large result set may consist of thousands of small information items or just a few large object blocks.

Commercial EAI products like IBM's Data Joiner [19] or Attunity's Connect [1] as well as some research prototypes like TSIMMIS [12] focus on integrating a previously known fixed set of data sources and developing an inte-

[1] supported by the German Research Foundation (DFG) within the Center of Excellence (SFB) 627

Table 1: Classification of other Systems

	Dynamic	Schema	Embed-ding costs	Integration mode	Data Sources	Result Size
EAI Systems	low	global	medium	schema-level, global as view	~10	~MB
Peer-to-Peer Systems	medium-high	local	(high)	schema-level, peer mappings	~10	~MB
Information Manifold	medium	global	medium	instance-level, local as view	~10^3	~MB
Semantic Mediation	medium	global, but extensible	medium	instance-level, as concept in ontology	~10-10^2	~MB
WWW/Search Engines	(high)	none	low	unified access, selection from keyword	~10^9	~kB
Semantic Web	(high)	glob. base, loc. refinement	low	unified access, selection from concept	~10^9	~kB
Grid	medium	none	low	unified access, selection from property	~10^3	~GB
Nexus	medium	global, but extensible	medium	instance-level, local as view	~10^6	~MB

grated schema for these sources. The area of schema integration [2] as well as software systems for this purpose, e.g. CLIO [14, 23], have a similarly restricted focus as they address the problem of transforming a fixed set of source schemas (and associated data) into an optimally calculated or predefined target schema. They lack the flexibility to handle dynamically changing sets of data providers and they do not aim at semantically integrating the data of different providers. Adding a new data source typically requires the intervention of an operator, and often leads to loss of information that does not fit in the predefined global schema. EAI and schema integration software can be used locally to transform the data of a provider into a common domain schema compliant representation.

Recently, new approaches for mediation systems tailored to Peer-to-Peer architectures have been proposed [5, 15]. Peer-to-Peer systems are highly dynamic, allowing for ad-hoc addition and removal of data sources. Traditional data integration systems with a fixed, global schema are not adequate for such architectures. Instead, the newly proposed architectures couple the peers directly and map their local schemas via domain relations and coordination formulas [5] or peer mappings [15]. An application directly sends queries to a single peer using the local schema of this peer, which may access other peers via the mappings to answer the request.

To connect n peers, between $O(n)$ and $O(n^2)$ mappings are necessary. Using fewer mappings reduces the costs for integrating a new data source, but leads to more schema conversions applied to a single piece of information on its way from its original source to the client. Schema conversions possibly involve some loss of information. No fully automatic way is provided to derive the mappings.

While a global schema indeed has some limitations in dynamic environments, there are still advantages. It provides a uniform interface for applications and reduces the number of mappings required to integrate a new data source. Moreover, in our approach, we focus on a single application domain, which simplifies the design of a global schema. We will show that the global schema approach can be adapted to dynamic environments by allowing subsequent extensions to the schema.

Information Manifold [22] aims at combining data from multiple sources using joins. Similar to our approach, they use queries over the global schema to describe the sources' contents, and unique identifiers to relate multiple representations of the same real-world object. However, they do not semantically merge the representations, and they do not consider extensions to the global schema. They enumerate all combinations of suitable sources, and they use the response of one source as the input for the next source, which we consider to be inefficient with many sources. In our IMMM, a query to a relevant source does not depend on the responses of other relevant sources.

Semantic Mediation [13] introduces a Mediator as transparency tier that hides the distribution of information providers from the application. The Context Specification Language defines mappings between source ontologies and the mediator ontology. These are used to integrate new sources with an existing mediator. Ontologies comprise both schema and instance concepts, thus the ontology remains only at a reasonable size within very restricted domains. Query processing involves reasoning along the ontology graph which we consider inefficient in general.

The World Wide Web and web search engines are today's most common distributed information systems. They currently do not offer schema information or semantics, which in consequence limits queries to simple keyword searches. As the indexes of search engines are kept up-to-date by web crawlers, there are usually large delays between addition and removal of a data source (web server) and the corresponding index update, leading to invalid references to data sources in search results.

The Semantic Web [4] is the next generation World Wide Web that contains machine-understandable information in addition to content meant to be read by humans. This enables search engines to retrieve information in a much more intelligent way as today. Technically, the Semantic Web is a set of portable standards to represent ontologies. Research in this area aims at developing languages to represent ontologies [10, 16, 27] and processing mechanisms. Inference machines reason about entities in one ontology. They also assist in ontology mapping by inferring relationships between entities of two ontologies.

The Semantic Web only deals with metadata. In contrast to our proposed middleware, it does not process or transform data or strive for providing a large-scale information integration infrastructure, but can be seen as a basis for one. It aims at providing one global solution for reasoning about all kinds of information. In contrast to that, our middleware focusses on efficient data integration and query processing in a delimited universe of discourse.

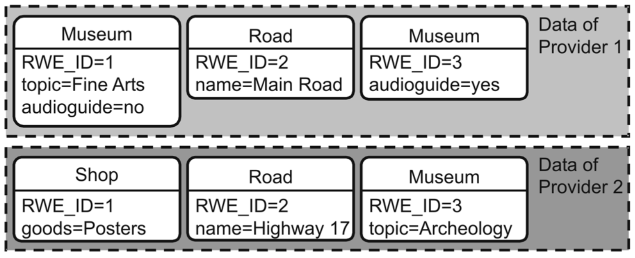

Figure 1: Overlapping data objects of two providers

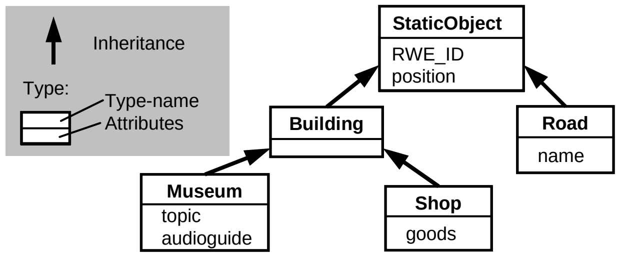

Figure 2: Simplified excerpt of a domain schema

A Grid [11] is a distributed computing infrastructure consisting of large and diverse sets of distributed resources and services. Virtualization services [26] are used to implement different levels of transparency between applications and the data sources or services. A discovery service provides location transparency and can be searched by applications for suitable entries by mapping logical domains and predicates onto actual data sources or services. In Data Grids, many small and large services, spread over multiple loosely coupled organizations, work together to provide access to and management of massive amounts of distributed data [17]. A main focus of the research are replica management and consistency issues. In most cases, the granularity of the content is a (possibly very large) file. Applications do not send complex queries over various data sources but use the Grid to find one data source that has the file they need. In [18], the discovery service is a registry where content providers store a "content link" (URL) to their data source and some metadata (e.g. mimetype). Each data object has to be retrieved by its own. There is no support for data integration or merging.

Table 1 summarizes the discussion of related work. Where results are bracketed, a definite assessment was not possible, see the discussion above.

3. Requisites for an IMMM

An information maximization mediation middleware (IMMM) allows applications to access data on real world entities as if it were stored in a single, large database. In this section we delimit the solution space by describing our assumptions on the data model and architecture used in such a middleware system, and highlight the impacts of the application domain on the design of the system.

3.1 Data Model Requisites

The data running through an IMMM is structured into units of information which we call objects. Each object represents a thing (an entity) in the real world consisting of a type and some attribute-value-pairs. Figure 1 shows examples of objects stored at two different providers. The first object of Provider 1 has the type Museum and has values assigned to RWE_ID, topic, and audioguide.

The term object refers to an instance of a type. The type of an object determines its valid set of attributes. A domain schema is the collection of all types of objects relevant to one universe of discourse – the application domain. There, types are organized in a type hierarchy such that each type, except for the root type, has a base type all of whose attributes it inherits. Figure 2 shows an excerpt of a domain schema for the application domain of location-based services (LBS), which are a special case of context-aware applications. StaticObject is the root type and Road and Building inherit from it. Valid attributes of the Road type are RWE_ID, position, and name. Types do not have methods as we focus on data integration.

Definition 1: Simple Object. *An object is simple, iff it has exactly one type and at most one value for each of its attributes.*

In Figure 1 all six objects are simple, but represent only three real world entities. Later in this section we detail on how to detect such multiple representations. Obviously, an application would prefer a merged representation of the result consisting of only one combined object per represented real world entity, as shown in Figure 3.

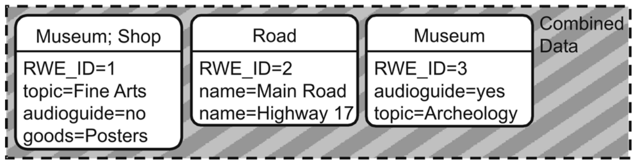

Figure 3: Result set of information request

Definition 2: Combined Object. *An object is combined, iff it contains all data (types and attributes) of two or more simple objects referring to the same real world entity.*

The motivation for combining objects is as follows. We focus on scenarios where "exact" results are not required, i.e. applications can cope with "best-effort" results containing all currently available data. So, in the example above, if only one provider is available, the application still gets some data (e.g. upper or lower half of Figure 1). If both providers are available it gets more details (see Figure 3). The idea behind merging multiple representations is to keep query results simple and to free the application from eliminating duplicates, while preserving all available information.

As a consequence of this, combined objects have two special characteristics: multi-type and multi-attribute.

Definition 3: Multi-type. *A multi-typed object is an object with more than one type. Valid attributes of a multi-typed object are those defined by any of its types.*

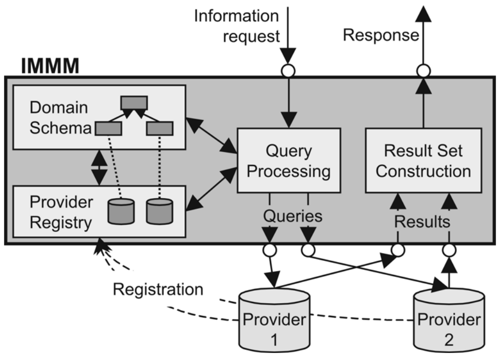

Figure 4: Architecture of an IMMM

In Figure 3 the first object has two types: Museum and Shop. It is probably a museum with an associated merchandising shop. By returning the combined object the application can draw a single icon on the map and still display all associated information like the exhibition topic or the shop's range of goods. Alternatively the application can draw two icons such that both are visible.

Definition 4: Multi-attribute. *Attributes having the same name can be grouped into a multi-attribute of the same name comprising their unique values.*

In Figure 3 the road object has two names, which are equally valid. Highway 17 is useful to a user rushing through the city, Main Road is more suitable for someone looking for a particular building in this road. By supplying both names the application can pick one based on the user's context or display both and let the user decide.

Identifying multiple representations. Detecting that two simple objects refer to the same real world entity is a difficult task that requires knowledge from the application domain and the definition of an equality metrics. In the LBS domain, a simple approach is to consider two simple objects to be matching if their position attributes are less than a threshold distance apart. More complex approaches are subject to ongoing research [7]; they also consider n-to-m matches between groups of simple objects (like roads) from two data sets. In this paper, we use a simple but effective aid: we consider two simple objects to be matching if their RWE_ID attribute has the same value. This requires a provider to look for matching objects (using semi-automatic tools) and to copy their IDs before publishing objects to the IMMM. We consider this a reasonable starting point, since it distributes the identification task to the data providers that typically know the detailed semantics of their data objects best.

3.2 Architecture Requisites

Figure 4 sketches the architecture of an IMMM. It consists of four main components: Domain Schema, Provider Registry, Query Processing, and Result Set Construction.

The *Domain Schema* contains knowledge about the types and valid attributes of objects used in the application domain of this IMMM. Other components use it to check if requests or objects comply with the schema.

In order to provide interoperability between providers and applications (a provider serves many applications, and an application uses many providers) the Domain Schema needs to be globally fixed. We detail in Section 5 on the benefits of this limitation. Providers may define extensions to this schema. Applications have to know them to benefit from the extended data. The Domain Schema component can be used to relate different extensions.

The *Provider Registry* determines the relevant providers for an information request issued by a user. Therefore, it stores a "table of contents" of each provider, which consists of the value ranges of the global index attributes of all objects of a provider. Providers need to register at the Provider Registry to introduce themselves to the IMMM.

Definition 5: Global index attributes. *Every information request contains constraints on the global index attributes to narrow down the result set. They are fundamentally important in an application domain.*

In the LBS domain these attributes are position and type, while in a car sales scenario they might be make and model.

The *Query Processing* component forwards all incoming requests to the relevant providers.

The *Result Set Construction* component merges the result sets of different providers into a single, combined result set. Simple objects are merged into combined objects as described in Section 3.1.

Application Domain. Applications can take advantage of an IMMM in various domains. In the neuro science domain [13], and similarly in bio and life sciences, many companies store data (e.g. research results and well-known knowledge) about cells, molecules, proteins, etc. in their private repositories, each one using its own schema. In car sales [22] several data providers offer different yet related data on used cars, luxury cars, and car reviews. In LBS numerous providers offer map data, data on hotels and restaurants, sightseeing and shopping guides, but they are not yet interoperable.

An IMMM can combine the data of all providers of a domain leading to a major benefit: applications can access the combined data in a flexible way. Thus, more versatile, more robust and more powerful applications founded on a bigger-than-ever data basis can leverage the combined knowledge of all providers to tackle more complex tasks.

To make this work, two conditions have to be met. First, a domain schema needs to be defined. Usually this is done by domain experts; in the LBS domain we have conducted a use case analysis to build such a schema [24], that makes use of schemas built by the OpenGIS Consortium for the domain of geo-spatial applications (e.g. [25]). Secondly, providers match their schema to the global schema and create a mapping between them. Typically, the data remains stored in the provider's schema, and a wrapper converts it to the global schema on-the-fly.

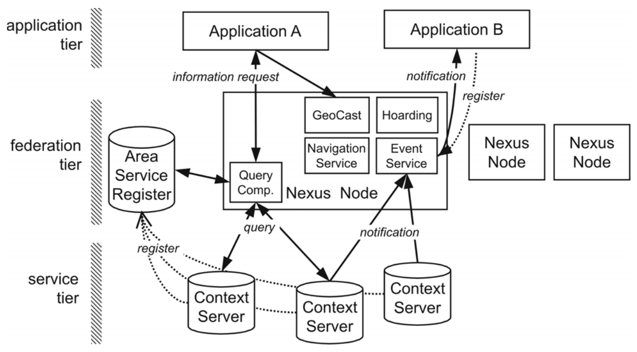

Figure 5: Architecture of the Nexus Platform

Impacts. The application domain has a substantial impact when deploying an IMMM. It determines the universe of discourse and defines the terms used there along with their semantics. The schema of an application domain builds upon those terms and semantics, and reflects the common understanding of the users of this domain about how to structure their information. Similarly, this common understanding leads to domain-specific global index attributes.

The semantics associated to types and attributes determine when to consider two objects as multiple representations of the same real-world entity. In some domains there exist algorithms [7] formalizing this implicit knowledge of how to find and merge multiple representations. For other domains the relation between multiple representations have to be determined manually and stored explicitly.

4. Implementation (Nexus)

In the Nexus Project a first prototype for an IMMM as outlined in the previous section has been built. It is tailored to the domain of context-aware applications. In this section we give an overview on the vision and the architecture of the Nexus Platform as shown in Figure 5 (more details are described in [24]).

4.1 The Mission

The goal of the Nexus Platform is to support all kinds of context-aware applications by providing a shared global context model. To achieve this goal, the platform federates local context models from so-called context servers. The local models contain different types of context information: representations of real world objects like streets, rooms, or persons and virtual objects that link to digital information spaces or that embody relationships between objects. Sensors keep the data of local context models up to date (e.g. the position of a person).

Many providers survey all kinds of data. However, this is an expensive and time-consuming effort, and therefore providers have a high interest in sharing their data to increase the revenue or to get other data in return. Thus, by leveraging our mediation middleware, providers can save a lot of work. Similarly, applications profit from the integrated data. They can just access information instead of having to look for it.

4.2 Context Server

A context server stores a local context model. It is comparable to a web server in the WWW. The server has to fulfill two requirements in order to be part of the Nexus Platform: it has to implement a certain interface (accepting simple spatial queries and returning results in a specified XML language) and it has to register with its service area and object types at the so-called Area Service Register (which is a spatially enabled Provider Registry). A context server can be implemented in various ways. For providing large scale geographical models, we use a spatially enabled database. We cope with the high update rates of the changing positions of mobile users using a distributed main memory system [21]. For the Aware Home [20] we adopted a lightweight spatial server as a context server. Even small-scaled sensor platforms like the ContextCube [3] can be used as a context server.

4.3 Federation

The federation – the query component in the Nexus node, to be precise – mediates between applications and context servers. It has the same interface as a context server, but does not store models (except for caching). Instead, it analyzes an application query, determines the context servers that may provide relevant data, and distributes the query to them. Then, it combines the incoming result sets to a consistent view and returns it to the application. For this, it employs its knowledge about the semantics of the objects and applies several semantic transformations on the result sets: merge multiple representations of a real world object, aggregate single buildings to a block of houses, or change the level of detail of a building's outline.

For query distribution and service discovery, a Nexus node uses the Area Service Register. This service is a directory of the available local context models. It stores the addresses of the context servers, their object types, and their spatial extent (the context server's service area). More details about the federation tier can be found in [24].

In addition to the query features, a Nexus node can support advanced value-added services which have their own interface and use the federated context model. Figure 5 shows some of them: With *GeoCast* [9], one can send a message to every mobile user that is currently in a given geographic region by addressing that area. *Hoarding* is the process of preloading data onto a mobile device to support disconnected operation [6].

4.4 Applications

A context-aware application can use the Nexus Platform in three different ways. First, it can send queries to the federation (query component) to get integrated information about its surroundings. Secondly, the application can register to the Event Service to receive a notification when a certain state of the world occurs, e.g. when the user has entered a building or the temperature in a room exceeds a certain threshold. Thirdly, it can use value-added services like the Navigation Service to shift the processing of common functions from its client to the infrastructure.

5. Conclusion

Our goal is to introduce a transparency tier that hides the distribution of data providers from the applications to achieve the impression of a single database. Thus, applications can benefit from new data providers without needing to change, and they are less affected from disappearing data providers. The difference between our approach and other approaches is that we explicitly do not look for generic solutions. We rather think that the restriction on a single application domain offers many advantages. In a single application domain it is possible to agree on a domain-specific global data schema, to use domain-specific algorithms, and even to cope with modeling details.

A global data schema allows information maximization, e.g., detection of duplicate objects, merging of multiple representations, generalization or aggregation of data. Furthermore, with a global data schema only a n:1 schema matching (every data provider to the global schema) is necessary, instead of n:m schema matchings (every provider to every other provider) in the worst case, or matchings using intermediate steps, as in P2P scenarios, where information loss is quite likely. In our vision, the global schema is comparable to an upper ontology, but in favor of efficiency it is restricted in some way. Therefore, we do not support inference techniques or reasoning processes in general, but we provide a domain-specific and limited query language. The limited query language, in turn, facilitates the efficient processing of queries in the federation, while it comprises all functionality needed by the domain's applications.

With these constraints, we can achieve better data quality and availability, and additionally offer value added services like spatial events. It is still a best effort approach – domain-specific integration architectures like the IMMM do not solve the problem of information integration. However they provide unique benefits over generic solutions.

6. References

[1] Attunity: *Attunity Connect Architecture,* White paper, March 2001, http://www.attunity.com/files/AttunityConnectTechWhitePaper.pdf

[2] C. Batini, M. Lenzerini, S.B. Navathe: *A Comparative Analysis of Methodologies for Database Schema Integration,* ACM Computing Surveys, Vol. 18, No. 4, Dec. 1986

[3] M. Bauer, C. Becker, J. Hähner, and G. Schiele: *ContextCube – Providing Context Information Ubiquitously,* Proc. of the 23rd Int'l. Conf. on Distributed Computing Systems Workshops, 2003

[4] T. Berners-Lee: *Weaving the Web,* Harpur, S. Francisco, '99

[5] P.A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, I. Zaihrayeu: *Data management for Peer-toPeer Computing: A Vision,* 5th Int'l. Workshop on the Web and Databases, Madison, Wisconsin, June 6-7, 2002

[6] S. Bürklen, P. Jose Marron and K. Rothermel: *An Enhanced Hoarding Approach on Graph Analysis,* Proc. of the Int'l. Conf. on Mobile Data Management (MDM), 2004

[7] T. Devogele: *A New Merging Process for Data Integration Based on the Discrete Fréchet Distance,* Proc. of the Joint Int'l. Symp. on Geospatial Theory, Processing and Applications, Ottawa, Canada, on CD, 2002

[8] Y. Ding and D. Fensel: *Ontology Library Systems: The Key to Successful Ontology Re-Use.* In Proc. 1st Int'l Semantic Web Working Symposium (SWWS'01), 2001

[9] F. Dürr and K. Rothermel: *On a Location Model for Fine-Grained GeoCast,* 5th Int'l. Conf. on Ubiquitous Computing, '03

[10] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, M. Klein: *OIL in a Nutshell,* Proc. of the 12th European Workshop on Knowledge Acquisition, Modeling, and Management, Springer Verlag, 2000

[11] I. Foster, C. Kesselmann, eds. *The Grid: Blueprint for a Future Computing Infrastructure,* Morgan Kaufman, 1999

[12] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, J. Widom: *The TSIMMIS Approach to Mediation: Data Models and Languages,* Journal of Intelligent Information Systems 8(2), '97

[13] A. Gupta, B. Ludäscher, M. Martone: *Registering Scientific Information Sources for Semantic Mediation,* Proc. of the 21st Int'l. Conf. on Conceptual Modeling, Tampere, Finland, 2002

[14] L.M. Haas, R.J. Miller, B. Niswonger, M. Tork Roth, P.M. Schwarz, E.L. Wimmers: *Transforming Heterogeneous Data with Database Middleware: Beyond Integration,* IEEE Data Engineering Bulletin, Vol. 22, No. 1, pp. 31-36, 1999

[15] A.Y. Halevy, Z.G. Ives, D. Suciu, I. Tatarinov: *Schema Mediation in Peer Data management Systems,* 19th Int'l. Conf. on Data Engineering, Bangalore, India, March 5-8, 2003

[16] J. Hendler, D. McGuinness: *The DARPA Agent Markup Language,* IEEE Intelligent Systems, Vol. 15, No. 6, 2000

[17] W. Hoschek, J. Jaen-Martinez, P. Kunszt, B. Segal, H. Stockinger, K. Stockinger, B. Tierney. *Data Management Architecture Report. Design, Requirements and Evaluation Criteria.* Technical report, DataGrid-02-D2.2, 2001.

[18] W. Hoschek. *A Database for Dynamic Distributed Content and its Application for Service and Resource Discovery.* In Int'l. IEEE Symp.on Parallel and Distributed Computing (ISPDC 2002), Iasi, Romania, July 2002

[19] IBM Corporation: *IBM DB2 Data Joiner,* Fact sheet, 2000, http://www-3.ibm.com/software/data/datajoiner/brochure/factsheet.pdf

[20] O. Lehmann, M. Bauer, C. Becker, and D. Nicklas: *From Home to World – Supporting Context-aware Applications through World Models,* Proc. of the 2nd IEEE Int'l. Conf. on Pervasive Computing and Communications, 2004

[21] A. Leonhardi and K. Rothermel: *Architecture of a large-scale location service,* 22nd Int'l. Conf. on Distributed Computing Systems (ICDCS), 2002

[22] A.Y. Levy, A. Rajaraman, J.J. Ordille: *Querying Heterogeneous Informaton Sources Using Source Descriptions,* 22nd Conf. on Very Large Databases, Mumbai (Bombay), India, 1996

[23] R.J. Miller, M.A. Hernández, L.M. Haas, L. Yan, C.T.H. Ho, R. Fagin, L. Pope: *The Clio Project: Managing Heterogeneity,* SIGMOD Rec. Vol. 30 Nr. 1, ACM Press, 2001

[24] D. Nicklas, M. Grossmann, T. Schwarz, S. Volz, B. Mitschang: *A Model-Based, Open Architecture for Mobile, Spatially Aware Applications,* Proc. of Symp. on Spatial and Temporal Databases, 2001

[25] OpenGIS Consortium (OGC): *Sensor Model Language for In-situ and Remote Sensors,* OGC discussion paper, OGC 02-026r4, Version 0.7, Dec. 2002, http://www.opengis.org/docs/02-026r4.pdf

[26] V. Raman, I. Narang, C. Crone, L. Haas, S. Malaika, T. Mukai, D. Wolfson, C. Baru. *Data Access and Management Services on Grid,* 5th Global Grid Forum (GGF5) Workshop, 2002, http://www.gridforum.org/Meetings/GGF5/papers.htm

[27] W3C: *OWL Web Ontology Language 1.0 Reference,* W3C Working Draft, 2002, http://www.w3.org/TR/2002/WD-owl-ref-20020729/