netzspannung.org
Wissensraum für digitale Kunst und Kultur

PAAL, Stefan
KAMMÜLLER, Reiner
FREISLEBEN, Bernd

Adaptable Web Interfaces for Heterogeneous Data Sources.

Publiziert auf netzspannung.org:
http://netzspannung.org/about/technology/
24.06.2004

Erstveröffentlichung**:** Proceedings of the 2nd Conference on Internet Computing (IC 2001). Las Vegas, USA. CSREA 2001. S. 827-834.

IMK
Fraunhofer Institut Medienkommunikation

The Exploratory Media Lab
MARS Media Arts & Research Studies

Adaptable Web Interfaces for Heterogeneous Data Sources

Stefan Paal
German National Research Center for Information Technology
St. Augustin, Germany

Reiner Kammüller
University of Siegen
Dept. of Computer Science
Siegen, Germany

Bernd Freisleben
University of Siegen
Dept. of Computer Science
Siegen, Germany

Abstract. *While the amount of information available over the internet is steadily growing, there is an increasing demand to efficiently and effectively make use of various shared information resources over the World Wide Web (WWW). Ideally, WWW applications should be shielded from the heterogeneous nature of underlying storage systems, and existing data sources should be transparently glued together by relying on a uniform WWW interface. In this paper, we present a new approach to develop a distributed storage system which achieves this goal for application users and additionally allows system developers to dynamically adapt the access to shared information to the needs of new applications. The proposed storage system is based on a new document-oriented storage approach which hides specific storage implementations from the applications using the data, while it is transparently extensible with new functionalities without breaking existing implementations. An example of its use is presented to demonstrate the feasibility of our proposal.*

Keywords: web interface; data sharing; heterogeneous environment; adaptable; database; distributed storage system

1. Introduction

Although we have used the internet for several years to share common knowledge, we are still unable to shape truly integrated World Wide Web (WWW) information systems. Particularly, their usage is often limited by providing access only to a single information space, not allowing to cross the borders of heterogeneous systems. Certainly, search engines like Google or Yahoo claim to break these limits, but they do not work on structured data nor can they access relational or object-oriented database systems (RDBMS or OODBMS). Furthermore, they do not offer an interface to access the information, rather they simply provide URLs for locating the resources.

In addition, information repositories are typically designed from scratch. This has led to databases obeying different sets of constraints to model the same information. Although one may potentially be able to access all available databases, in reality this is an almost impossible task due to the large variety of storage interfaces. A straightforward approach would be to define a common storage interface and to force all information providers to implement this interface. But this would limit the potential functionalities of accessing the data, loosing the advantages of high level semantic access to particular application-dependent data repositories.

Consequently, instead of defining a static storage interface which covers common access features to different databases, we should provide

a dynamic storage access system to integrate the data sources in a uniform way.

Due to the diversity of information and the various options to process and visualize it, another emerging demand is the customizable access to web repositories tailored to the specific needs of the user [1,2]. Therefore, different applications may be used concurrently, which should be able to access commonly shared data with specialized interfaces. The challenge is to give a user the illusion that he or she is accessing a commonly shared web knowledge space, containing various repositories beneath a uniform web interface tailored to his or her needs.

In this paper, we present an approach to cope with these problems and introduce a distributed storage system which is able to integrate various existing WWW data sources to form a shared information system and offers possibilities to the developers to access different data sources using storage adapters. Furthermore, the proposed storage system is open enough to be extended with new data sources without breaking existing implementations. The underlying idea of this approach is to introduce a new document oriented storage architecture, wherein data and the related methods are no longer separated and the applications can benefit from a high-level API, hiding the specific storage implementation.

The paper is organized as follows. In section 2, we discuss features of web information systems, especially regarding the integration of different data sources. Section 3 presents our approach in detail, followed by section 4 with a description of the implementation. Section 5 concludes the paper and outlines areas of future work.

2. Web Information Systems

In this section, we briefly discuss the relevant issues of web information systems with respect to distributed storage systems. The discussion is based treating the demands of users and developers separately.

The user of a web information system expects an implementation which can be tailored to his or her specific needs, triggered by himself or herself. On the other hand, a user does not want to be bothered by changes of the underlying implementation details like database types or network connections. Thus, the used storage system should be able to operate independently of the source and format of the used data, and the composed parts should be able to interact dynamically, forming new storage components.

The developer expects from a web information system to be open enough for various applications. There should be support to implement new features and to extend the existing system, rather than implementing a new one. Additionally, the use of standards regarding tools, programming languages, network protocols or database types eases the acceptance through the developers and shortens the development time. On the other hand, runtime support is needed with respect to the adaptation of the system. The system should also provide the possibility to plug in components dynamically. Common features, such as handling of access control, should be portable between the components and the environment has also to be future-proof and flexible enough to integrate new extensions without breaking existing implementations.

Consequently, we summarize the objectives of a web information system with respect to the underlying data sources as follows:

- The storage system cannot be implemented as a monolithic system, rather it must be composed from individual components.
- The storage system must provide interfaces to access these components and their functions in a uniform way, which do not rely on component specific features, but also allow the components to introduce more specialized functions.
- The storage system should be dynamically extensible with components providing access to new application-specific data formats or documents, integrating other storage systems as well as introducing new, document orthogonal functionalities like concurrency handling, awareness services, access control.

There are several approaches [3,4] to solve the adaptation problem and to provide extensibility and composability of storage systems in this context with techniques like *open implementations* or *reflection* [5] However, these approaches offer only partial solutions with respect to the objectives defined above. For example, [6] addresses flexible data sharing

within groupware systems, but does not deal with heterogeneous environments. Other approaches like Enterprise Java Beans [7] are limited to particular languages like Java or C++ [8,9], network protocols [10,11], database environments [12,13,14] and user interface [8,15], thus are not transferable to other scenarios. Furthermore, there are approaches which support the adaptation of information systems, but they do not solve the problem concerning separation of data and their corresponding operations [16,17]. To the best of our knowledge, there is no approach addressing all of the defined objectives.

3. A Distributed Storage System

In this section, we present the basic concepts of our distributed storage system and its features. At first, we outline the principles of different levels of data abstractions in commonly used system designs. In addition, we describe the basic approaches to share distributed data and compare their properties related to providing transparent access and their employment in heterogeneous environments. Afterwards, we illustrate how transparent access to various data sources can be achieved by introducing adapters, shifting the handling of low-level data structures from the application to the data sources. Therefore, applications providing web access do not have to deal with particular data source implementations in the internet.

3.1 Levels of Data Abstractions

The main architectural disadvantage of existing storage systems is the lack of exact knowledge about the data they store. Certainly, there are different types of storage systems for various applications, and they provide different approaches for storing data. However, many of them end at the logical representations of the data, i.e. they use the underlying physical data format like sectors on a hard disk and map their logical data format like relational tables on it. The given interfaces deal more or less perfectly with the access to this data format, providing various application-independent functions like access and concurrency control. This feature is on the one hand an advantage, because it is usable for different application types, on the other hand it is a disadvantage, since the application has to be adapted to the used storage system by mapping the specific document format again onto the logical representation, and by introducing a document interface as it is shown in Figure 1.

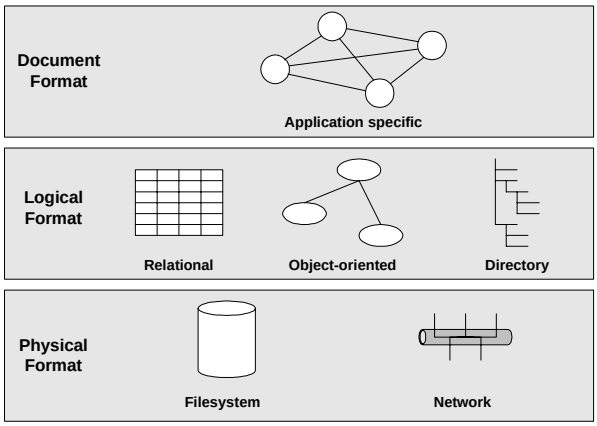

Figure 1: Levels of Data Abstraction

Although this approach works well for single, stand-alone applications, it fails in case of distributed, heterogeneous storage systems, which have to provide access for many application types and which in addition should be able to be extended dynamically by new logical formats. In this scenario, all applications sharing the same document format have to be adapted to use the specific logical data format of the current storage system. Furthermore, every change of the storage systems has a direct impact on the application implementation, thus making the whole system more statically than dynamically extensible.

3.2 Accessing Distributed Data in Heterogeneous Environments

The introduction of the two-layer architecture (Figure 2, left) with separated application logic and database server is the first step to improve the situation mentioned above, but does not overcome the basic problem. However, it can be used to provide distributed access to shared storage systems. Due to the lack of an application-independent way for adapting heterogeneous logical formats, existing implementations use the data sources as they are, resulting in a tight

binding with the specific data source. Commonly used database interfaces like ODBC or JDBC do not change anything about it, since they work on the logical format and inherit the disadvantages mentioned above.

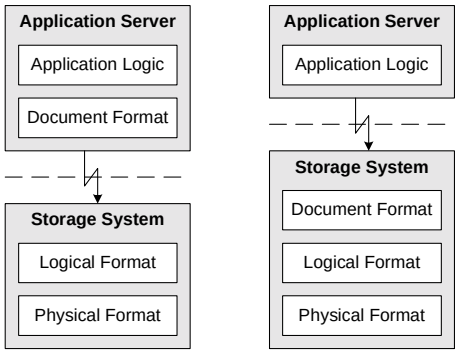

Figure 2: Storage and Document Oriented Access

Several approaches go a step further, enhancing the database server with additional storage components like Enterprise Java Beans [7] and providing access on a higher abstraction level, shifting the document format layer onto the database server (Figure 2, right). At first glance, the problems seem to be solved, but in fact, the provided solutions are intended for and thus restricted to limited application scenarios. Either they support only one single database type or are restricted to certain programming languages. Furthermore, they often cannot be extended and mixed transparently with other, existing storage systems. This is particularly important in an open, adaptable environment, which will be dynamically changed and extended.

3.3 Uniform Storage Access

Both users and developers of web information systems do not want to deal with different types of storage sources, rather they want to access the documents independent of where and how they are stored. To realize transparent and uniform access to heterogeneous data sources, we introduce the concept of a *storage adapter* which hides the implementation of the underlying data source (Figure 3, left). Thus, we can integrate different types of storage systems, and the application is able to access them in a transparent, uniform way. An example from one of our projects is the parallel use of an object-oriented database like Poet and a relational database like Postgres. Even if they have completely different approaches to store data in their logical format, they implement the commonly used document interface, enabling the application to access the data in the same way and hiding database specific implementations.

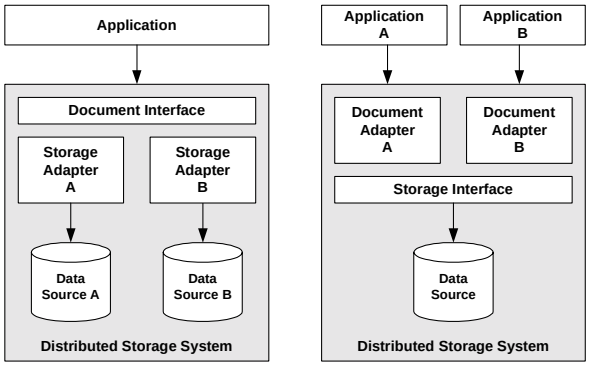

Figure 3: Uniform Storage and Document Access

The introduced approach is not only applicable to conventional storage systems like databases and file systems, but also for more abstract storage systems like network connections or any kind of data exchange. Consequently, the proposed distributed storage system represents a new abstraction which combines several other approaches. It is the foundation for composing a commonly shared information space, spanning different data sources.

3.4 Uniform Document Access

Since applications and document formats are changed over time, requirements are modified or new ones are added, but the stored data should still be accessible by all corresponding applications, we additionally introduce so called *document adapters* (Figure 3, right). They prepare the data access for each document type individually. For example, in one of our projects we have provided document adapters for proprietary access to the data as well as document adapters which convert the stored data on the fly to XML (Extended Markup Language). Thus, we can share the same data with different co-existing

application types, and each user can work with his or her preferred application. In addition, this approach also ensures that the user does not have to change the web application when he or she accesses another information system, rather the application can access different document formats in a uniform way.

3.5 Document-Oriented Approach

The usage of storage and document adapters does not only shield an application from the underlying storage systems and the evolution of document formats, it also offers a new document-oriented approach to extend the functionality transparently and simultaneously without breaking existing implementations (Figure 4).

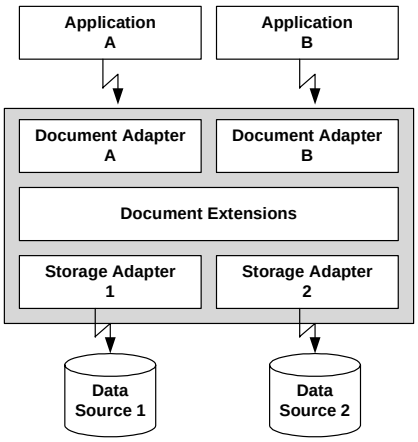

Figure 4: Document Oriented Approach

The basic idea of this approach is to introduce an additional layer between storage and document specific adaptation, where document extensions can be integrated transparently and reused between various versions of implementation. The extensions can enhance the current document interface with new specific features like awareness, but also with orthogonal aspects like access control or concurrency control.

4. Implementation Issues

After having illustrated the basic concepts of the distributed storage system and uniform data access, we present in this section an overview of the implementation which we have developed and used in several projects.

4.1 Overview

The logical architecture of the distributed storage system (Figure 5) mainly consists of three layers, which contain the document and storage adapters introduced in the previous section as well as the document handling layer. On the bottom, there are the different storage systems, storing the data in their physical data format and providing logical access. On the top, there are the applications accessing the stored data in a transparent and uniform way. For both, the distributed storage system acts as a glue, which connects different applications and storage systems. In particular, the adaptation and composition of new document handling features can be made dynamically, integrated in the document handling layer without breaking the application implementation.

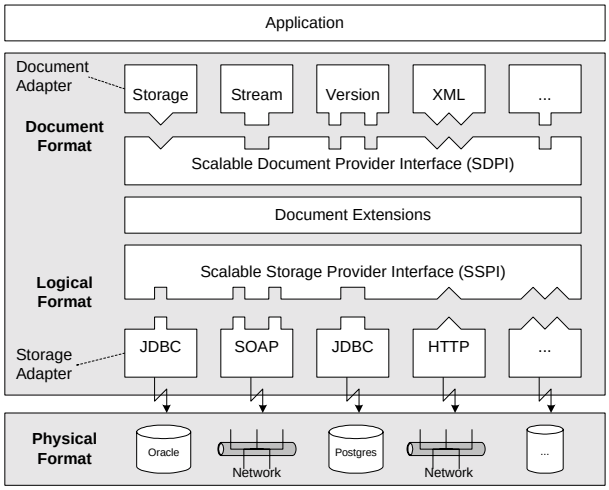

Figure 5: Distributed Storage System

We have implemented the presented system in C++ and also in Java, which both allow us to use object-oriented approaches, but differ in the support for creating a scalable, adaptable storage framework. In this context, the main problem to solve is the adaptation of different existing interfaces in a way that the caller and the called method do not have to be changed, since they also often cannot be modified. For this purpose, a

software design pattern, the *Object Adapter* pattern or a variation of it, the *Proxy* Pattern, is applied, which takes the method call from the caller and delegates it further to the right implemented method. In addition, the adapter could modify the method call, e.g. by changing calling parameters or even could call other methods [18].

Since the software development started some years ago, there were no tools to automate the application of these patterns. We were forced to integrate the patterns into our source code, thus dealing manually with many software layers and cluttering the source code with recurrent portions of adapter code. The introduction of Aspect-Oriented Programming (AOP) and related tools described in [19] offer an approach to automate this in an elegant manner. Instead of mixing the source code for handling the actual storage system with the adapter and proxy code, the development can easily be separated. Even more, the same common functionalities, also called *aspects*, such as access or concurrency control, can be reused for different storage systems. However, although AOP is an interesting approach for adapting existing source code, it is applied during software development time. This limits its usage for scenarios where the source code is available. Currently, there are several works in progress to overcome these limitations, especially in Java environments using the language inherent reflection techniques [3].

4.2 Example

The project we want to present is called CAT (Communication, Art & Technology) [20], which is intended to build up the upcoming information, communication and production platform for art, culture and new media in Germany, called *netzspannung.org* [21]. It is funded by the German Federal Ministry for Education and Research and will be developed by the research group IMK/MARS from GMD, St. Augustin in cooperation with the University of Siegen, Germany. The CAT platform is heavily based on a distributed system especially developed for connecting open communities spread over the internet in a new way. The members of these communities have no longer to rely on rigid structures with given network architectures, protocols and data formats, but rather they want to be able to build a community tailored to their specific needs.

For this purpose, the platform can be extended dynamically with self-defined modules and data storage systems, which are spread into the community network or stored on a dedicated machine at the member's home. In addition, *netzspannung.org* can be seen as a glue for connecting various existing resources like huge databases, search engines and internet portals. This could happen in a flexible manner from the database level up to the user interface, depending on the desired degree of interaction. Therefore, users as well as developers of *netzspannung.org* benefit greatly from the proposed distributed storage system.

Figure 6 outlines how an application, in this case an implementation of a web-based forum, can benefit from the presented storage system.

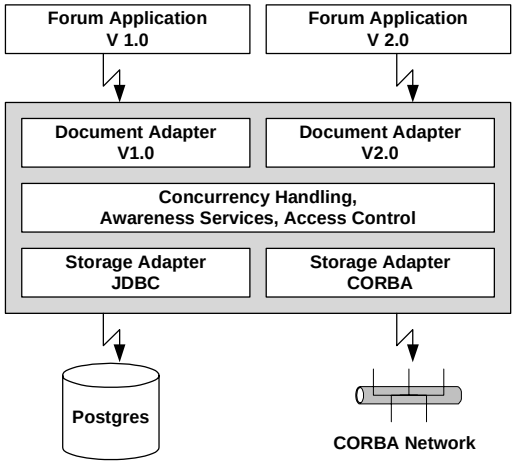

Figure 6: Integrating Different Data Sources and Document Formats

Firstly, we have to integrate transparently various data sources like the object-relational database Postgres and a CORBA network connection. Both are linked to the system by two specialized storage adapters, therefore shielding the system from storage specific properties. Secondly, the internet platform *netzspannung.org* is an open community system which offers members the ability to adapt interfaces and the way documents are processed. This leads to

different forum implementations, illustrated in figure 6 by V1.0 and V2.0, which are composed of concurrently operating modules. The task of the storage system is to achieve this without breaking existing implementations, but also to offer simultaneous access to commonly shared data. Thirdly, the implementation benefits from the document oriented approach, because additional functionalities like concurrency handling, awareness services and access control can be integrated transparently and mainly reused within different versions. They are attached on the document level itself and not on the storage or the application specific implementation level. Both sides, the application as well as the data sources, are detached from the document extension layer by the introduced adapters.

5. Conclusions and Outlook

In this paper, we have presented a distributed storage system which is intended but not limited to support the specific document and storage requirements of web information systems. It can be extended dynamically with new functionalities using adapters without breaking existing implementations. We have briefly presented an example of an open distributed community system, where in conjunction with Java, every user can extend and adapt the available data sources and document formats to his or her specific needs, accessing commonly shared data in uniform way.

There are several areas for future research. For example, after XML (Extensible Markup Language) and SOAP (Simple Object Access Protocol) have emerged, we have started to work on dynamic adapters, which do not modify the interfaces like in AOP [5], but transform the data transmitted in a method call between two objects using SOAP. A related approach is proposed in [22], which introduces so called intermediaries to manipulate information streams and which are applied during runtime. Thus, in contrast to the conventional Adapter Pattern, the adaptation can be conducted even if no source code is available. This will be a step towards a freely customizable information space.

In conjunction with XML, several concepts have been emerged about using and evaluating semantics in database systems [23]. The goal is to make data access more efficient by not only evaluating the content of resources, but also their meanings. We currently investigate how to provide database independent access to meta data, similar to the introduction of document adapters.

Some additional work has already been started on using the storage system not only for server based information spaces, but also in widely distributed peer-to-peer networks. In this case, the challenge is to transform the internet from a low-structured hyperlink system to a commonly shared knowledge repository, building the next internet generation.

References

[1] Catarci, T. Web-based Information Access. International Conference on Cooperative Information Systems. IEEE 1999. pp. 10-20.

[2] Montebello, M. Wrapping WWW Information Sources. International Symposium on Database Engineering and Applications. IEEE 2000. pp. 431-436.

[3] Seiter, L., Mezini, M., and Lieberherr, K. Dynamic Component Gluing in Java. Proc. of 1st Symposium on Generative and Component-Based Software Engineering (GCSE '99), LNCS. Springer 1999.

[4] Mezini, M., Seiter, L., and K. Lieberherr. Component Integration with Pluggable Composite Adapters. Kluwer Academic Publications, 2000.

[5] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Videira Lopes C., Loingtier, J.-M., and Irwin, J. Aspect-Oriented Programming. Proc. of the European Conference on Object-Oriented Programming (ECOOP), Finland. Springer-Verlag LNCS 1241. June 1997.

[6] O'Grady, T. Flexible Data Sharing in a Groupware Toolkit. M.Sc. thesis, Dept. of Computer Science, University of Calgary, Calgary, Alberta, Canada. 1996.

[7] Monson-Haefel, R. Enterprise JavaBeans. O'Reilly & Associates. 2000.

[8] Goeschka, K.M., Falb, J., Radinger, W. Database Access with HTML and Java - A Comparison Based on Practical Experiences. Proc. on the 22nd Annual International Computer Software and Applications Conference (COMPSAC '98). IEEE 1998. pp. 588-593.

[9] Bouguettaya, A., Benatallah, B., Hendra, L., Ouzzani, M., Beard, J. Supporting Dynamic Interactions Among Web-based Information Sources. IEEE Transactions on Knowledge and Data Engineering. IEEE 2000. pp. 779-801.

[10] Benatallah, B., Bouguettaya, A. Data Sharing on the Web. First International Workshop on Enterprise Distributed Object Computing Workshop. IEEE 1997. pp. 258-269.

[11] Wang, N., Chen, Y., Yu, B., and Wang, N. Versatile: A Scalable CORBA-based System for Integrating Distributed Data. International Conference on Intelligent Processing Systems. IEEE 1997. pp. 1589-1593.

[12] Petrou, C., Hadjiefthymiades, S., Martakos, D. An XML-based, 3-Tier Scheme for Integrating Heterogeneous Information Sources to the WWW. Proc. on the 10th Database and Expert Systems Applications (DEXA 99). IEEE 1999. pp. 706-710.

[13] Kappel, G., Kapsammer, E., Rausch-Schott, R., and Retschitzegger. X-Ray - Towards Integrating XML and Relational Database Systems. Proc. of the 19th International Conference on Conceptual Modeling, (ER'2000). Salt Lake City, USA, October, 2000.

[14] Leontiev, Y., Ozsu, M. T., and Szafron, D. On Separation between Interface, Implementation, and Representation in Object DBMSs. Proc. of the Technology of Object-Oriented Languages and Systems, 1998. pp. 155-167.

[15] Dridi, F., Neumann, G. How to Implement Web-Based Groupware Systems Based on WebDav. Proc. of the 8th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. IEEE 1998. pp. 114-119.

[16] Härder, T., Loeser, H., Zhang, N. Supporting Adaptable Technical Information Systems in Heterogeneous Environments - Using WWW and ORDBMS -, in: Proc. 8th Int. Workshop on Database and Expert Systems Applications (DEXA'97), Toulouse, Sept. 1997, pp. 295-303.

[17] Hergula, K., Härder, T.: A Middleware Approach for Combining Heterogeneous Data Sources - Integration of Generic Query and Predefined Function Access, in: Proc. 1st Int. Conf. on Web Information Systems Engineering (WISE 2000), Hongkong, June 2000, pp. 22-29.

[18] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley. 1995.

[19] Tatsubori, M., and Chiba, S. Programming Support of Design Patterns with Compile-Time Reflection. OOPSLA'98 Workshop on Reflective Programming in C++ and Java, pp.56-60, Vancouver, Canada, Oct 18, 1998.

[20] Fleischmann, M., Strauss, W. Communication of Art and Technology (CAT). IMK/MARS, GMD St. Augustin. http://imk.gmd.de/images/mars/files/Band_1_download.pdf

[21] netzspannung.org, Communication Platform for Digital Art and Media Culture. http://netzspannung.org

[22] Barrett, R., Maglio, P. P. Intermediaries: An Approach to Manipulating Information Streams. IBM Systems Journal, 38, 1999. pp. 629-641.

[23] Morgenstern, M. Integrating Web and Database Information for Collaboration Through Explicit Metadata. Proc. of the Seventh International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. IEEE 1998. pp. 204-210.