

# Active XML Database Architecture

V. Krylov  
MERA Labs  
Nizhny Novgorod,  
603163, Russia  
+78312722020

vkrylov@meralabs.com

D. Ponomarev  
MERA Group  
Nizhny Novgorod,  
603163, Russia  
+78312788800

dmp@meranetworks.com

A. Logvinov  
MERA Labs  
Nizhny Novgorod,  
603163, Russia  
+78312788865+3125

alogvinov@meralabs.com

A. Ponomarenko  
MERA Labs  
Nizhny Novgorod,  
603163, Russia  
+78312788865+3126

aponom@meralabs.com

## ABSTRACT

We introduce the active XML database architecture to build very large, scalable, loosely structured distributed data storage.

Traditionally data is regarded as passive records operated by DBMS software. Our idea is that every data unit is active, capable of communication with other data units and database clients. Combined with the special overlay structure incrementally formed by data units (Metriized Small World Graph) this provides for effective distribution of data units among database servers and unbounded scalability of the resulting storage ensuring logarithmic search and append complexity.

Each active data unit is represented as an XML document addressable by a unique URL having a locally stored extendable set of XLink links to other data units, and a software module driving the communication with other data units and clients. Search in this structure is performed by sequential and/or parallel crawling following the links in the list obtained on each step.

The active data units communicate by sending XML messages over a transport protocol such as HTTP. The communication includes the retrieval of XML content and link lists, addition of new links, calculating query relevance and work delegation (so that every unit can actively propagate the process initiated or mediated by another unit).

Since there are no central controlling nodes in the structure, multiple processes of adding new data units and searching for existing data can be performed independently and simultaneously, and begin with any existing data unit. Moreover, because the data units are active, these processes may propagate on they own without being fully dictated by originator. This allows the distribution of data processing along with the distribution of data itself.

We have built a prototype implementation of the architecture. The analysis of the small world overlay structure properties confirmed the possibility of building efficient XML data storages which contain hundreds of petabytes of data.

## 1. INTRODUCTION

The traditional approach to building databases can be characterized by the following statements:

1. Data items are considered passive entities operated by the DB management software.
2. Data items are stored in structures which generally have a single entry point and a sequential process of finding the relevant data.

But modern applications set the requirements which make this model not optimal, namely the requirement to store and process large amount of data generated by the steadily increasing number of uncoordinated sources having no common control center and queried by numerous and uncoordinated data consumers. To fulfill this requirement, new data storage architectures are required with the following properties:

1. Decentralization of both data and data access, since the sources of data and queries are uncoordinated and geographically distributed.
2. Accessibility of data in relatively short time compared to the large volume of data in the storage.
3. Scalability, because the amount of data and query sources grows constantly.
4. The ability to work with weakly structured data, since imposing restrictions on the structure of data elements generated by numerous uncoordinated sources is impractical.

In the Related Works section we give an overview of the existing work in the area of building distributed and decentralized data storages. The Active Data Unit section describes the proposed method of data item representation which provides for decentralization and distribution of both data and processes pertaining to the data. The Metriized Small World Overlay Structure section describes the way to connect data items which allows for finding the relevant data in relatively short time even when number of data items is huge and constantly growing, permitting a multitude of search and addition processes occurring at the same moment of time. In the Processes section we describe how the aforementioned processes are performed. Finally, the Simulation section gives provides analysis of the properties of Metriized Small World Overlay structure.

## 2. ACTIVE DATA UNITS

Each active data unit (ADU) consists of the following components:

1. XML content.
2. List of XLink links to other ADUs.
3. Message processing module (the active component).

Each of these components has a unique URI which is a sub-URI of the base URI of the active data unit, so that if the URI of any of components is known, the URIs of other components can be trivially calculated. Links in the list (2) point to the base URIs of other ADUs.

ADUs communicate with each other and with non-ADU actors (such as database clients) by means of XML messages. Each message contains the following information:

1. Message identifier which determines the semantics of processing of the message.
2. List of string parameters which further specify the details of processing.
3. List of attached XML documents involved in the processing of the message.

Messages are exchanged in request-response manner, which means that after a message is processed, the response message is returned to the sender of the original message. If HTTP is used as transport protocol, the original message would be sent in an HTTP POST request and the response message would be received in the HTTP response. The latter means that non-ADU agents are not required to have URIs in order to send messages to and obtain responses from ADUs, but cannot receive messages other than responses to their own messages.

Active data units use the resources of servers which host them and their base URIs are sub-URIs of the server base URI. Therefore at the network level the storage can be viewed as a set of servers processing ADUs and exchanging messages pertaining to those ADUs. But ultimately the ADUs, not the servers, are the endpoints of communication; hence the distribution of ADUs between servers is irrelevant to the semantics of the communication, influencing only the time which is necessary for the messages to reach the destination. The communication of ADUs processed by the same server is logically indistinguishable from the communication of ADUs processed by different servers but can be significantly faster because the network is not involved.

The following operations are possible on ADUs:

1. Retrieve XML content of the ADU.
2. Retrieve the list of links of the ADU.
3. Add a link to the list of links of the ADU.
4. Calculate the metric value between the XML content of the ADU and a given XML document (the metric values are described in the section 3).

Each operation is initiated by sending the message with the corresponding identifier to the target ADU. The result of the operation is retrieved in the response message.

### 3. OVERLAY METRIZED SMALL WORLD STRUCTURE

The overlay Metrized Small World graph structure (MSW) is formed by the ADUs, which represent the vertices of the graph, and by the links locally stored in each ADU which these represent the edges of the graph. For every link (A, B) stored in ADU A,

there is a reverse link (B, A) stored in the ADU B, so the MSW graph can be considered undirected. No ADU can contain a link to itself.

To ensure that the overlay structure formed by ADUs has the properties listed in the Introduction section, the following requirements must be met:

1. Any ADU in the structure must be able to serve as an entry point of access to other elements. This is necessary for decentralization since having fixed entry points would create a bottleneck for data access.
2. A relatively short path must exist between every two ADUs to ensure search in a small number of operations compared to the number of ADUs.
3. Any ADU must store a relatively small number of links so that the access to individual links will not present storage and search complexity problems in itself.

There are also requirements pertaining to the method of construction of the structure rather than to the structure itself:

1. The structure must be formed incrementally with the addition of new ADUs, i.e. the addition of a new ADU must not require reconfiguration of the links between previously added ADUs.
2. The addition of a new ADU must not require iteration over the whole set or a large subset of ADUs.

Nowadays the structures are known which satisfy the three requirements above: the Small World graphs [15]. It was noted that the Small World graphs have the following properties:

1. Power law vertex degree distribution.
2. The average shortest path length between two vertices proportional to the logarithm of the number of vertices.
3. The clustering coefficient independent of the number of vertices.

The existing methods of construction of Small World graphs described in [16] are not suitable because they fail to satisfy the second structure formation requirement above by requiring the iteration over the whole set of vertices each time a new vertex is added. There is also a way to reorganize the data in a peer to peer system to establish the small world properties described in [17], but it fails both requirements on the construction of structure by requiring both iteration over the whole set of links and rewiring of the existing links. The dynamic incremental method Small World graph construction in application to MSW structure which satisfies both requirements is described in the Processes section.

The Small World graph structure is the necessary prerequisite for the effective search since it ensures the existence of a short sequence of links between any two ADUs. But there also must be a way to find this sequence without trying any possible direction on each step, i.e. there must be some useful measure of proximity of an ADU to the search pattern which will govern the choice of the next link on each step. Furthermore, a proximity measure between ADUs is necessary for construction of the structure in order to make the relative position of the elements consistent with the search algorithm, i.e. to create useful pattern-to-ADU proximity gradients and to clusterize the similar ADUs. The latter means that ADUs which are measured as close to each other are separated by a smaller than average number of links.

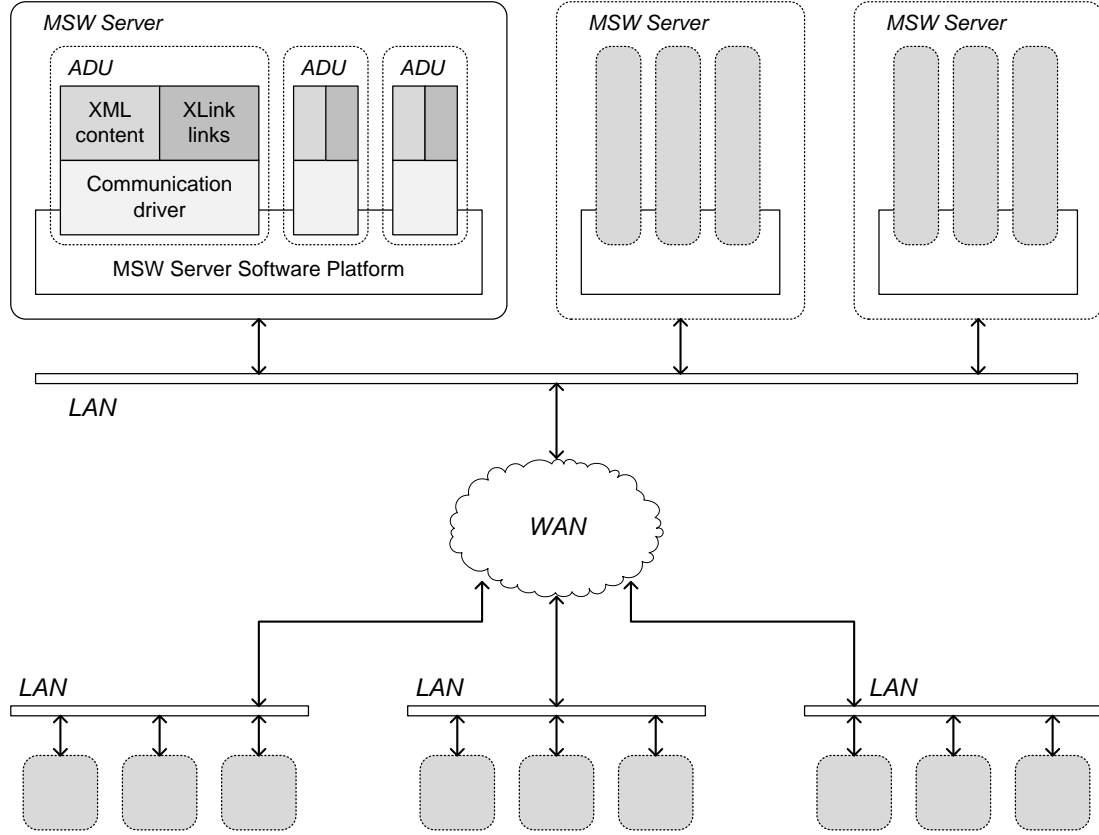


Figure 1. Active distributed MSW XML database architecture.

We propose the unified proximity measure between ADUs and between the search pattern and an ADU which we call the semi-metric. The semi metric is calculated between two arbitrary XML documents, i.e. either between the content of two ADUs or between the content of an ADU and the search pattern. The search pattern is considered to have the same structure as ADUs which are expected to match it with relevant XML elements or attributes set to specified values and irrelevant XML elements omitted from the pattern.

Figure 1 gives an overview of the proposed architecture on the component level. Multiple ADUs are hosted by each MSW server sharing the common server software platform. The servers are connected by a LAN allowing multiple servers per a single local area network. Ultimately each server can connect to any other server via WAN.

#### 4. PROCESSES

There three possible processes in the MSW structure:

1. Searching for existing ADUs.
2. Adding new ADUs.
3. Modifying existing ADUs.

The third process is performed by marking the ADU being modified as obsolete and creating a new version of that ADU with different URI and modified data. The ADUs marked as obsolete will not appear in search results but still can serve as intermediate points in a search process. There is also the possibility to use these obsolete ADUs to query the data pertaining to the particular

moment in the past by returning obsolete versions of ADUs instead of the actual ones if the actual ones were added after the specified moment of time.

Assuming that the search process can be reduced to set operations on the sets of results of searching ADUs which are relevant to a set of patterns, we will further consider the search process as process of finding ADUs relevant to a single search pattern.

The search process in the MSW storage starts from an arbitrary selected ADU (the entry point) and continues by following the links obtained on each step until the relevant ADU is reached or the time or step number quota is exhausted. If more than one relevant ADU is expected, the process can be branched after acquiring the first result ADU to obtain the entire set or a subset of relevant results. The relevance of an ADU to the query can be expressed as a function of the semi-metric value between the ADU and the query.

We propose the following algorithm of adding new ADUs to the existing MSW structure, which is also the algorithm by which the MSW structure is incrementally constructed. In this algorithm  $n$  is the number of algorithm steps per single addition (determines MSW characteristics preservation),  $m$  is the number of links established by the ADU being added,  $n \geq m$ .

Let's assume that the structure already contains  $i - 1$  ADUs and we want to add the  $i$ -th ADU. Then the algorithm is as follows:

1. Arbitrarily select an ADU  $v_k$  where  $1 \leq k \leq i - 1$ .
2. Let VisitedList be the set of visited ADUs.

3. Let CandidateList be the set of candidate ADUs for link establishment sorted by value of semi-metric to  $v_i$  in ascending order.
4. Assume that CandidateLists initially contains only  $v_k$ .
5. For  $j < -1$  to  $n$  do
  - a. Sort CandidateList by value of semi-metric to  $v_i$  in ascending order.
  - b. Select the first ADU  $p$  from CandidateList not contained in VisitedList. If no such ADU exists then break.
  - c. Add  $p$  to VisitedList.
  - d. Add the set of  $p$  neighbor ADUs to CandidateList.
6. Mutually connect the  $v_i$  ADU with  $m$  arbitrary ADUs from VisitedList.

One important consequence of the decentralization of the MSW structure is that processes in it can be performed entirely independently and simultaneously since they are unlikely to involve the same ADUs. If the process  $P1$  involves the set of ADUs  $A1$  and the process  $P2$  involves the set of ADUs  $A2$ , they can occur totally asynchronous (i.e. not causing each other to wait for a common resource) if  $A1 \cap A2 = \emptyset$ . If  $A1 \cap A2 = A_c \neq \emptyset$ , a separate synchronization lock will be required on each ADU from  $A_c$ , assuming the worst case of ADUs processing messages sequentially. This will only cause either  $P1$  or  $P2$  to wait for a lock if they happen to access the same element from  $A_c$  simultaneously.

The processes in the MSW database can be driven either externally or internally. In the former case a database client communicates individually with each ADU involved in the process, making the decision on each step regarding how the process will continue. The ADUs only respond to the messages described in the Active Data Units section without initiating any messages themselves. In this approach the client has full control over the addition and search processes, which has the disadvantage of the consistence of the MSW structure being

dependent on the compliance of each client with the addition algorithm described above.

The internal approach to the processes control consists in creation of a special ADU containing the search pattern which does not participate in the MSW structure, but communicates with the ADUs of the MSW structure to obtain search results in the same way as a client would communicate with ADUs in the external approach. The communication of the search pattern ADU and the client can be arranged as follows:

1. The client creates the search pattern ADU  $A_s$ .
2. The client sends the search initiation message  $M_{si}$  to  $A_s$  and waits until  $A_s$  responds.
3.  $A_s$  performs the search process and sends the response message  $M_{sr}$  back to the client containing the set  $S_r$  of URIs of the result ADUs in the form of an attached XML document containing XLink links to each result ADUs.
4. The client reads the result set  $S_r$  from the  $M_{sr}$  message and retrieves the contents of the result ADUs.

The addition process using the internal approach is performed in a similar way, but the acting entity is the ADU being added.

The advantage of the internal approach is that all actions of the ADUs are controlled by the server software as opposed to the client software in the external approach. Therefore the responsibility for MSW structure consistency lies on servers, not on clients, which is preferable from the security and reliability point of view.

## 5. SIMULATION

We provide the results obtained from execution of the proposed ADU addition algorithm using XML media item descriptions as test data. Structures containing 1000, 5000, 10000 and 20000 elements were assembled using this algorithm. The graphs of vertex degree and shortest path length distributions are shown in Figures 2-5. In Figure 6 you can see how the average shortest path length between vertices changes with the increasing number of vertices in the structure.

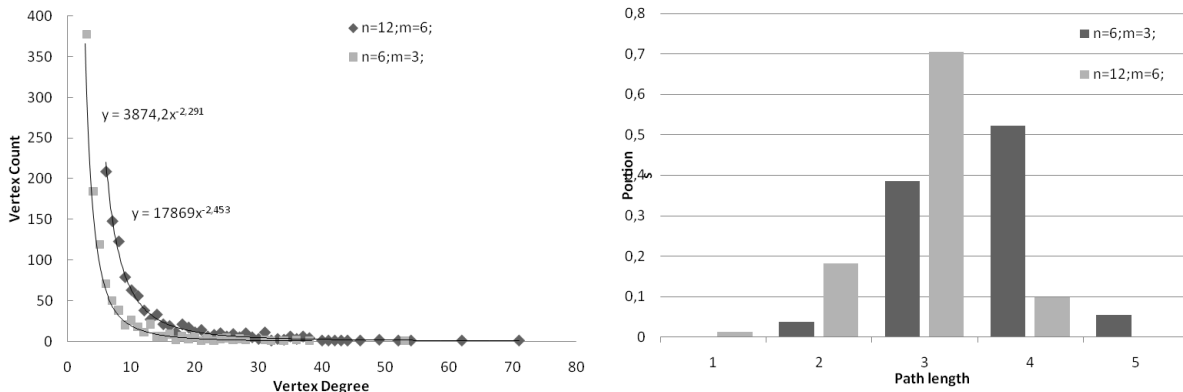
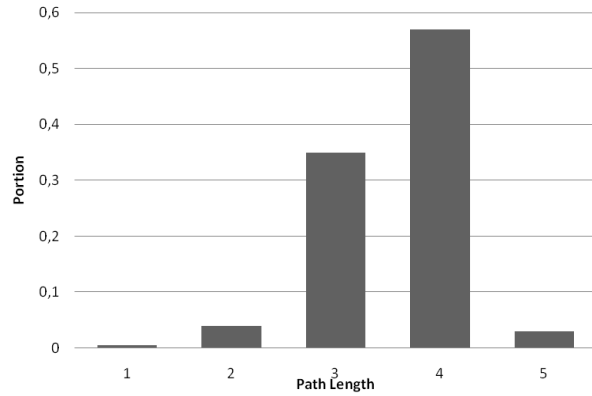
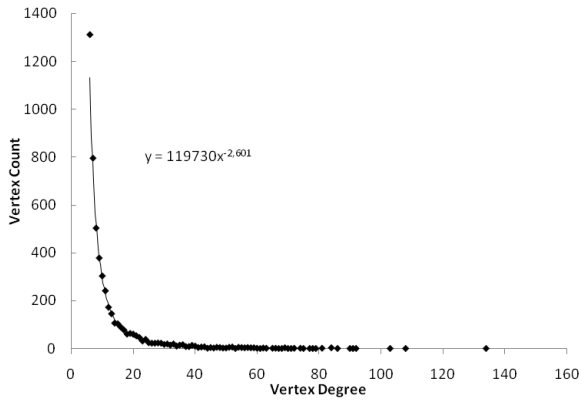
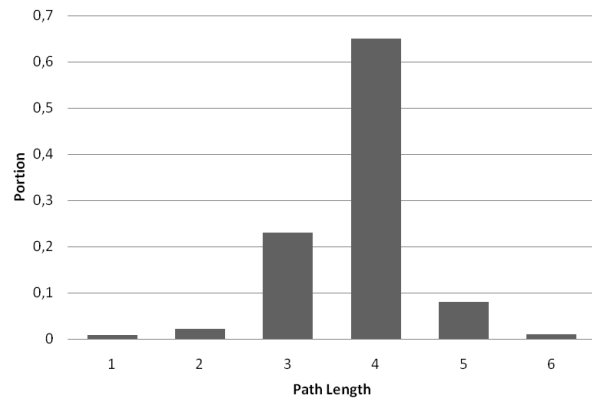
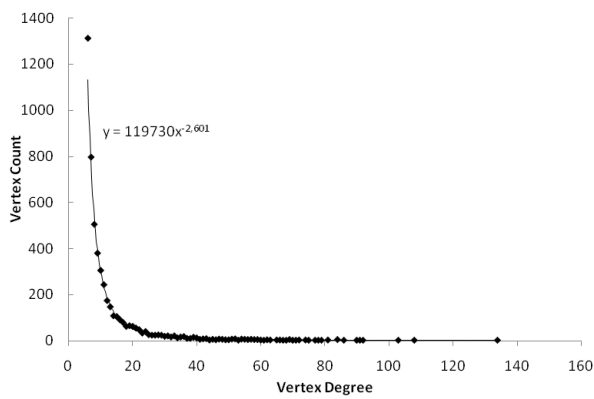


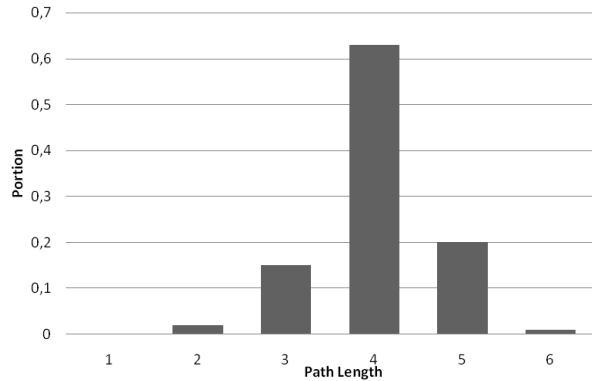
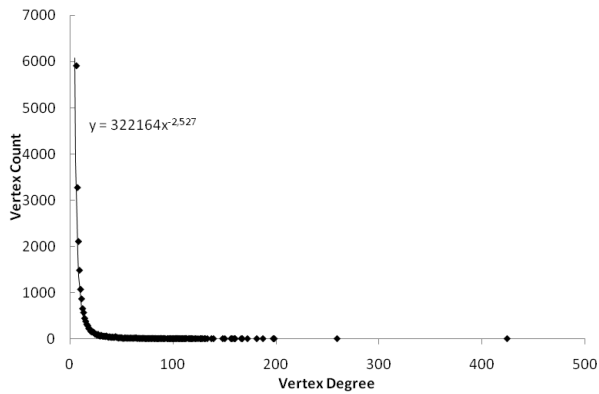
Figure 2. Vertex degree distribution and shortest path length distribution. The number of vertices in the structure is 1000.



**Figure 3. Vertexes degree distribution and shortest path length distribution. The number of vertices in the structure is 5000.**



**Figure 4: Vertex degree distribution and shortest path length distribution. The number of vertices in the structure is 10000.**



**Figure 5. Vertex degree distribution and shortest path length distribution. The number of vertices in the structure is 20000.**

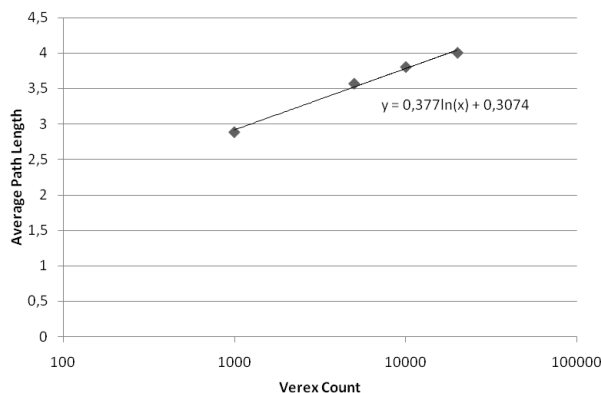


Figure 6. Shortest path length distribution averaged over all structures.

## 6. RELATED WORK

One of the key goals of the proposed XML database architecture is to provide means of creating very large, scalable databases that require little to none administration. In [1] authors describe a zero-administration system for sharing and querying XML data. The system allows users to share XML data without significant human intervention, and to pose XQuery FLWR queries against them. Unlike the proposed architecture, XPeer implements a hybrid p2p model, in which super-peer nodes coexist with simple peer nodes. Super-peers are chosen on voluntary basis (usually these are nodes with adequate computational power and/or network bandwidth) and perform administration tasks.

In the proposed architecture both data and data access are decentralized to enable effective data querying, since the sources of data and queries are uncoordinated and geographically distributed. In [2] authors describe a distributed database extension called XRPC. The authors' strategy for advancing the state-of-the-art is to incrementally extend functionality of stand-alone XML database systems to P2P in all database related dimensions (including query execution, query optimization, transaction management and data storage) by developing a working P2P XDBMS prototype as a test-bed for their research and to work on applications that benefit from P2P database technologies. XRPC enhances the existing concept of XQuery functions with the Remote Procedure Call (RPC) paradigm.

In [3] authors describe the Piazza system, a peer data management system for XML data, which provides an infrastructure for building Semantic Web applications. The architecture of Piazza is basically a hierarchical p2p architecture, where peers are fully autonomous, and may contribute data with schemas, while a central node hosts an index structure for query routing and performs query reformulation.

One of the key goals of Piazza is to provide semantic mediation between disparate data sources. Federated databases [4] and data integration systems [5, 6] both address this problem, but they rely on a two-tier mediator architecture, in which data sources are mapped to a global mediated schema that encompassed all available information.

In [7] authors describe DBGlobe, a p2p system for global computing. The project's key point is mobile peers' management; the peers may relocate over time. Other key points are the use of services for dealing with heterogeneity and mismatching

problems, as well as the use of Active XML [8] as the paradigm for service invocation/execution and data exchange. However, the DBGlobe system requires extensive administration and maintenance.

In [9] authors study query evaluation on Active XML documents (AXML). AXML documents are XML documents whose content is given partly by explicit data elements, and, partly, by embedded calls to Web services, which can be invoked to generate data. The authors' major challenge was the efficient evaluation of queries over such documents is to detect which calls may bring data that is relevant for the query execution. The authors formalize the problem, provide algorithms to solve it and present an implementation that is compliant with XML and Web services standards.

The work [10] considers the problem of distributing and replicating AXML data and services in a peer-to-peer setting. XML documents with embedded calls to Web Services are used in many different systems [11, 12, 13, 14].

## 7. REFERENCES

- [1] C. Sartiani, P. Manghi, G. Ghelli, and G. Conforti. XPeer: A Self-Organizing XML P2P Database System. In *P2P&DB*, 2004.
- [2] Zhang Y., Towards P2P XML Database Technology. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, Vienna, Austria, PhD Workshop, September 2007.
- [3] Halevy, A.Y., Ives, Z.G., Mork, P., Tatarinov, I.: Piazza: data management infrastructure for semantic web applications. In: *Proceedings of the Twelfth International World Wide Web Conference, WWW2003*, Budapest, Hungary, 20-24 May 2003, ACM (2003) 556-567
- [4] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183.236, 1990.
- [5] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *ICDE '95*, pages 251.260, 1995.
- [6] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB*, pages 251.262, Bombay, India, 1996.

- [7] Pitoura, E., Abiteboul, S., Pfoser, D., Samaras, G., Vazirgiannis, M.: DBGlobe: a service-oriented P2P system for global computing. *Sigmod record* 32 (2003) 77–82
- [8] Abiteboul, S., Benjelloun, O., Manolescu, I., Milo, T., Weber, R.: Active XML: Peer-to-Peer Data and Web Services Integration. In: *28th International Conference on Very Large Data Bases (VLDB 2002)*, Hong Kong, China, August 20-23, 2002, Proceedings, Morgan Kaufmann (2002) 1087–1090
- [9] Serge Abiteboul, Omar Benjelloun, Bogdan Cautis, Ioana Manolescu, Tova Milo, Nicoleta Preda: Lazy Query Evaluation for Active XML, *SIGMOD 2004*
- [10] S. Abiteboul, A. Bonifati, G. Cobena, I. Manolescu, and T. Milo. Dynamic XML documents with distribution and replication. In *Proc. of ACM SIGMOD*, 2003.
- [11] J. Powell and T. Maxwell. Integrating Office XP Smart Tags with the Microsoft .NET Platform. <http://msdn.microsoft.com>.
- [12] Jelly: Executable XML. <http://jakarta.apache.org/commons/sandbox/jelly>.
- [13] Macromedia Coldfusion MX. <http://www.macromedia.com/software/coldfusion/>.
- [14] S. Abiteboul, O. Benjelloun, I. Manolescu, T. Milo, and R. Weber. Active XML: Peer-to-peer data and web services integration (demo). In *Proc. of VLDB*, 2002.
- [15] R. Albert and A.-L. Barabasi. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74(1): 47-97, January 2002.
- [16] F. C. W. Aiello and L. Lu. Random Evolution in massive graphs. In *Proceedings of the 42<sup>nd</sup> IEEE symposium on Foundations of Computer Science*, page 510. FOCS, IEEE Computer Society, 2001.
- [17] C. Schmitz. Self-organisation of a small world by topic. In *Proc. Of the 1st Intl. Workshop on Peer-to-Peer Knowledge Management*, pages 61-66, 2005.