Searching Shared Content in Communities with the Data Ring

Serge Abiteboul*
INRIA-Saclay
Serge.Abiteboul@inria.fr

Neoklis Polyzotis†
Univ. of California Santa Cruz
alkis@ucsc.edu

Abstract

Information ubiquity has created a large crowd of users (most notably scientists), who could employ DBMS technology to share and search their data more effectively. Still, this user base prefers to keep its data in files that can be easily managed by applications such as spreadsheets, rather than deal with the complexity and rigidity of modern database systems.

In this article, we describe a vision for enabling non-experts, such as scientists, to build content sharing communities *in a true database fashion: declaratively. The proposed infrastructure, called the* data ring*, enables users to share and search their data with minimal effort; the user points to the data that should be shared, and the data ring becomes responsible for automatically indexing the data (to make it accessible), replicating it (for availability), and reorganizing its physical storage (for better query performance). We outline the salient features of our proposal, and outline recent technical advancements in realizing data rings.*

1 Introduction

Imagine a community of scientists who collect experimental data. It is common within such communities to share data in order to promote more research and enable the quick dissemination of information. (SWISSPROT[1] and the Genome Browser[2] are characteristic examples of this scenario.) In turn, content sharing necessitates a service for searching and/or querying, to enable the discovery of useful information out of the collected data. These two elements of sharing and searching can be found in other contexts as well, especially with the Web and the creation of user communities, such as Flickr. This evidence indicates an emerging trend towards building what we call *content sharing communities*: groups of users that wish to share and query information in some specific domain.

In principle, a distributed DBMS provides all the services needed to support content sharing communities. In practice, however, users avoid using DBMS technology due to its complexity. For instance, the conceptually simple operation of loading data in a DBMS involves several tasks, such as, defining a schema, tuning the

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*S. Abiteboul is a member of LRI, University Paris Sud and LSV, ENS Cachan. His work was partially supported by the ERC Grant Webdam and the ANR Grant Dataring.

†Part of the work was performed while the author was visiting INRIA, France.

[1] http://www.expasy.org/sprot/

[2] http://genome.ucsc.edu

database, collecting statistics, etc., that are not trivial for non-experts. These tasks are dwarfed in complexity by the next logical step, which is linking the local databases in a distributed or federated system. And even if a database expert is found to perform all these time consuming tasks, users may still find a conventional database system too rigid when it comes to its control over the data. Essentially, data has to be imported in the DBMS before it can be indexed and searched, and it has to be exported before it can be processed by other software.

We believe that the aforementioned user base forms a formidable challenge for database researchers and a chance to bring database systems to the "masses". In this direction, we propose the *data ring* that can be seen as a network analogue of a database or a content warehouse. The vision is to build a P2P middleware system that can be used by a community of non-experts, such as scientists, to build content sharing communities in a declarative fashion. Essentially, a peer joins a data ring by designating which files (or services in general) are to be shared, without having to specify a schema for the data, load it in a store, create any indices on it, or specify anything complex regarding its distribution. The data ring enables users to perform searches or declarative queries over the aggregated data, and becomes responsible for reorganizing the physical storage of data and for controlling its distribution. Thus a primary focus of the data ring is *simplicity of use*. The inherent diversity and complexity is hidden behind a simple and unified interface for publishing, querying, monitoring, integrating, and updating the available information. Besides clicking to select resources, the use of a ring resource should be no more complicated than the use of the same resource in a familiar local system.

As previously stressed, data rings target non-expert users, implying that the deployment of the ring and its administration should be almost effort-less. The "pay-as-you-go" philosophy (promoted by "dataspaces" [11]) is replaced here by "pay-only-by-providing-content"; the physical organization is (almost) for free. A first essential facet of the data ring proposal is thus the use of fully automatic and distributed data administration, both for data residing in file systems (a most common situation) and for more controlled systems, such as databases. Access structures, e.g., materialized views or indexes, are automatically introduced by the system when needed. Also, the system may choose to "activate" static data residing in the file system, e.g. a large collection of RSS seeds, to assign its management to a more efficient system such as a relational DBMS. The relational system is then in charge of managing the collection and providing efficient mechanisms for querying and monitoring.

The remainder of the article briefly reviews the salient features of the data ring proposal, insisting on its most novel and challenging aspects. We also discuss recent technical advancements in realizing a distributed data catalog that supports the search functionality of the data ring system. For both parts, the complete details can be found in the respective publications [3, 4].

2 Related Work

In principle, a data sharing community can be maintained as a distributed database using existing systems. Under this model, each user runs a local DBMS on the data that he/she wishes to publish, and all the participating databases are connected in a loose federation that forms a virtual global repository. As mentioned earlier, the main issue of this approach is the increased complexity of setting up and tuning such a system. It should be noted that modern database systems are equipped with several "advisors" (e.g., [5]) that can assist users in the difficult task of system maintenance. Such tools, however, implicitly assume that the user has some experience with database systems and can thus make informed decisions based on the recommendations that the advisors generate. Moreover, the proposed techniques focus primarily on the tuning of a centralized system, and it is not clear if they can be extended to the equally important problem of tuning the distributed system.

Peer-to-Peer (P2P) file sharing systems, such as, Gnutella or eDonkey, offer a light-weight alternative to the complexity of a full blown database system. Their main shortcoming is that they allow the location of files based solely on name matching, whereas users in data sharing communities are primarily interested in locating data based on content. Moreover, P2P file-sharing systems typically rely on query flooding as the main query processing mechanism, which does not scale well for complex queries and high querying rates.

Several research projects in academia have investigated the development of P2P database systems as the platform for supporting data sharing communities. In particular, previous works have investigated issues related to data integration [20], system design [9], and processing of complex queries [15, 12] over P2P networks. These aspects are undoubtedly important in the development of tools for the creation of data sharing communities. As noted earlier, however, an important issue is the development of mechanisms for self-administration that allow the P2P system to operate efficiently in an autonomous fashion. In this direction, earlier studies [13, 10] have investigated self-tuning techniques for the DHT [19, 18] substrate of P2P systems. These works are clearly an important step in the realization of self-managed distributed systems. It is equally important, of course, to explore self-tuning mechanisms for the upper layers of the system, that provide the support for complex queries over the distributed data.

The data ring proposal borrows ideas from several previous works on distributed data management. From P2P content warehouse [1], we adopt the idea of using semantic tools to enrich the data at our disposal, e.g. by discovering links between pieces of data. From [11], we borrow the vision of a *dataspace* where data is integrated. Optimization techniques can be used from the existing large body of works on distributed query optimization, such as PIER [15] and ActiveXML [6]. In particular, our proposal resembles the goals of PIER [15] in that it introduces a middleware system for running distributed query processing applications. The main difference is that we focus on a system to be used by non-experts, insisting on declarative management wherever possible.

3 The Data Ring: Design Overview

At an abstract level, a data ring is formed by a collection of peers, where each peer exports a set of resources (services and data). The collaborating peers are autonomous, heterogeneous, and their capabilities may greatly vary, e.g., from a sensor to a large database. Moreover, their exported resources may be quite diverse, e.g., some peers enable access to data resources, such as, a phone book or a genome bank, while others offer computational resources, such as, an ontology-based classifier, or a gene matching library. In what follows, we use P to denote a peer in the data ring, and $r@P$ to denote a resource r that peer P exports. A resource r can be either a data item or a service that P hosts.

We advocate a data model based on ActiveXML in order to represent services and data in a unified framework. More specifically, we assume that each data item is represented with an XML view, while services are specified in the Web Services Description Language [8] (WSDL). Two observations are in order. First, we employ XML solely as a model for data integration, and do not insist that each peer uses XML as its native data model. We assume that the latter is determined on a per-peer basis, depending on the type of exported data. Second, we advocate an enriched XML model that can capture both extensional and intensional data, i.e., data obtained through web service calls.

Peers can query the shared information using tree-pattern queries. We choose this formalism since it forms the basis of the structured FLWR construct of XQuery, and also because it is flexible enough to express unstructured search queries based on keywords and element tags. Figure 1 shows two example tree-pattern queries corresponding to these cases. A tree-pattern query issued at some peer is forwarded, though the data ring, to peers containing relevant data, where it is translated locally to a native query over the published resources. Therefore, the data ring borrows several features from mediation systems [16].

Following the established principles of database systems, a data ring organizes information at three levels of abstraction: the external layer, the topological layer, and the physical layer. (See Figure 2). The following summarizes the functionality of each part.

Topological Layer The topological layer contains the logical description of the information managed by the data ring, i.e., the set of participating peers and the resources that they export. While the topological

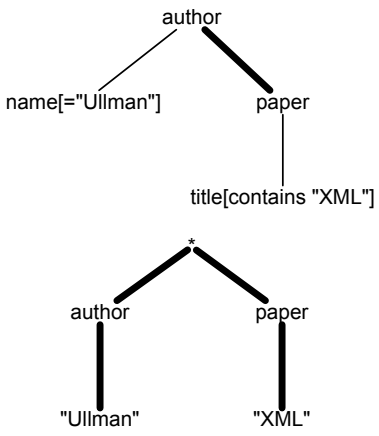

Figure 1: Two example tree-pattern queries over a repository of AXML documents with bibliographic publications. The top query applies specific structural constraints on the matched elements, whereas the bottom query only specifies containment predicates without restricting the structure.

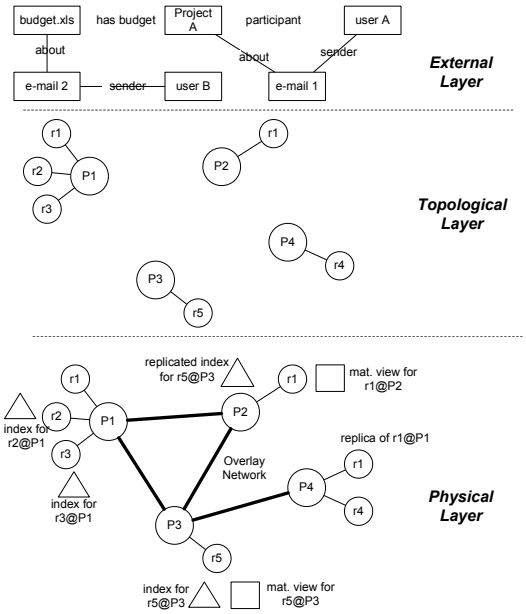

Figure 2: Information abstraction in a data ring.

layer does not expose the overlay network that is used to realize the distributed system, it does include information on the location[3] of resources, i.e., the $@P$ specification.

The topological layer provides declarative query services, in the same way that the logical model of a relational database supports SQL queries. A peer can thus submit an ad-hoc query against the registered resources, monitor a resource for changes, or open a continuous query against a stream resource (e.g., sensor readings, or an RSS stream). We also assume that the topological layer records the lineage and uncertainty properties of the data, and supports their integration in the query language.

External Layer As described previously, the topological layer manages different kinds of information: traditional data (as in relational systems), content (mails, letters, reports, etc.), metadata, as well as domain specific knowledge (e.g., ontologies) needed to interpret data and metadata. Since this level of detail can still be daunting for a large class of users, we enable the generation and maintenance of semantically richer data models in the *external layer*. For instance, the external layer can house the data models that are proposed in dataspaces [11]. In this fashion, a user will observe concepts and relationships between concepts, and will pose semantically rich search queries such as "what is the name of John Doe's company", typically using some simple syntax or some forms-based interface. We expect such queries to be realized by combining declarative queries over the topological layer, driven by the semantic information in the external data model.

Physical Layer The physical layer supports the evaluation of distributed query plans over the registered resources. It comprises a physical model and language for distributed query evaluation, as well as physical structures for managing the registered data. In particular, the physical layer provides a DHT service through some overlay network [19], on top of which the data ring maintains a catalog of meta data on the published resources. The meta-data includes the location of resources, information on their con-

[3]This is why we use "topological" instead of the more common term "logical" for the identification of the layer.

tents/capabilities, statistics on data and workload distribution, and in general any information that is pertinent to the evaluation of distributed queries. In general, we distinguish two key features of the physical layer: (a) it is completely distributed, and (b) it is self-administrated. These two elements combined introduce novel challenges in the design of autonomic systems.

Our work in the data ring project focuses mostly on the topological and physical layers, which we discuss further in the coming sections. This is not to say that we find the external layer to be less important; on the contrary! We hope to find some synergy here with works in dataspace support platforms [11], which examine data models similar to the ones that we envision for the external layer.

4 Distributed Catalog

As mentioned earlier, the physical layer maintains a catalog of meta data for the published content. Among other services, the catalog provides the following functionality that is relevant for our discussion: Given a tree-pattern query Q, it returns a super-set of peers that publish content relevant to Q. The idea is that a peer searches for content by first consulting the catalog with the corresponding query Q and then forwarding Q to the returned peers. Accordingly, a peer publishes resources in the data ring by pushing the corresponding meta data to the catalog. Overall, the catalog can be viewed as a distributed index of the published content.

For the purpose of indexing, each element in a published AXML document is identified by a structural identifier $sid = (start, end, lev)$. Here, $start$ (resp. end) is the number assigned to the opening (resp. closing) tag of the element, when reading the document and numbering its tags in the order they appear in the document. The third value lev denotes the element's level in the tree. Structural ids allow deciding if element e_1 is an ancestor of element e_2 by verifying if $e_1.start < e_2.start < e_1.end$. If $e_1.lev = e_2.lev$ holds in addition, then e_1 is the parent of e_2.

The catalog indexes element labels as well as words[4] in documents. We henceforth use *term* to refer to either of the two. The indexing scheme is based on the *Term* relation, defined as follows:

Term(p,d,sid,l) l is the label of element (p, d, sid)
Term(p,d,sid,w) w is a word under element (p, d, sid)

We refer to a tuple in *Term* as a *posting*. Given a term a, we refer to the set of its postings as the *posting list* for a and denote it as L_a.

In the data ring, XML documents are stored at their publishing peer, whereas the *Term* relation is distributed among the peers of the system using the DHT service of the physical layer [19, 18]. The keys of the DHT entries are the term values, and thus each peer becomes in charge of a set of posting lists. We use $Term_p$ to denote the horizontal fragment of *Term* that is stored in peer p. Given a query Q, the data ring retrieves the posting list L_a for each tag a that appears in Q and performs a parallel holistic twig join [7] to identify matching elements. The structural ids of the matching elements point to the peers to which Q is forwarded for further processing.

We developed two optimizations in order to improve the efficiency of the catalog, namely, distributed posting partitioning and structural bloom filters. Distributed posting partitioning (DPP for short) addresses the performance problems resulting from the high skew in the distribution of posting list sizes. The idea is to split the posting list for a popular term horizontally in *blocks* based on range conditions on structural identifiers, and to migrate portions to other peers. This leads to a hierarchical organization of the posting lists in the style of distributed B-trees [17]. Thus, a request for the posting list L_a is satisfied by doing parallel transfers for the blocks of L_a, which leads to higher efficiency. The holistic twig join is also modified to take into account the new organization. Specifically, the algorithm examines the range conditions that define the split of L_a in order to prune blocks that provably do not join with blocks from other lists.

[4] For efficiency, stop words are excluded from the index.

The second optimization of *Structural Bloom Filters* (SBFs for short) is orthogonal to DPP and aims to reduce the total volume of transferred data for the twig join. A SBF provides a compact representation of a set of postings that is suitable for filtering the postings of another list. As an example, assume that the join operator at the query peer P checks the descendant condition $a//b$ between the posting lists L_a and L_b located at peers P_a and P_b respectively. SBFs enable the following strategy for avoiding the transfer of the complete lists: P_a computes locally a small SBF F_a that summarizes L_a; P_b receives F_a and uses it to select a sublist $L'_a \subseteq L_a$ that provably contains all the b postings that are descendants of a; finally, the join operator at P performs the join between L_a and L'_b. Thus, the idea is to perform some local computation and exchange a small filter in order to avoid sending b postings that are provably not in the result of the twig join. Similar to bloom filters for relational data, SBFs commit only false positive errors with a bounded probability that can be tuned. We also develop several query processing strategies that utilize SBFs and that offer different trade-offs depending on the characteristics of the tree pattern query. The choice of the optimal strategy is a distributed query optimization problem with interesting research questions.

5 Stream Calculus & Algebra

An essential aspect of our proposal is the seamless transition between explicit and intentional data, which is needed in order to support effectively the loose integration paradigm of the data ring. ActiveXML [6] was designed to capture such issues and this is why we choose it as the foundation for the topological layer. An ActiveXML document is an XML document where certain elements denote embedded calls to Web services. For instance, the company document may contain the CEO phone number as a Web service call.

A language in the style of ActiveXML should also serve as the basis for the physical layer. In particular, this language should permit the use of functions in both pull and push (subscription) modes, the processing of streams of data, and of recursion (because of the graph nature of the Web). As shown in a recent work [2], distributed query evaluation and optimization can be naturally captured using ActiveXML algebraic expressions, based on the exchange of distributed query execution plans. The expressions include standard algebraic XML operations and send/receive operators, all over XML streams. Note that these may be seen as particular workflow descriptions of a strong database flavor.

An important element in this context is the cooperation between local query evaluation (under the responsibility of local systems, perhaps relational systems) and global query evaluation that is the domain of the data ring. The data ring sees the local query processors (with their optimizers) as boxes with possibly different querying capabilities, in the same vein as mediation systems [14].

6 Autonomic Administration

As mentioned from the beginning, our vision is to enable non-experts, such as scientists, to create content sharing communities with minimal overhead. In turn, this clearly implies that the users should be alleviated from any administration duties, and hence the ring should function without the intervention of an administrator or any central authority. Moreover, the distributed nature of data rings suggests that this autonomic operation must be completely decentralized.

Our goal of autonomic administration clearly targets the physical layer of the data ring. Similar to previous proposals on autonomic computing, we envision in particular the following functionality: self monitoring (e.g., gathering automatically statistics), self tuning (e.g., selecting automatically indexes), or self healing (e.g., selecting automatically a substitute for some failing Web service). The notion of self-administration and most importantly self-tuning have already been explored in the context of relational databases. However, in the data ring context they become an absolute necessity because of the nature of the system, and they also acquire unique characteristics: autonomic administration is continuously "on" so that it can reconfigure the system in

vu of volatile workloads; and, it requires the collaboration of the peers, whereas existing solutions on self-administration typically target a centralized scenario.

Clearly, the previous requirements set the bar very high. We believe that the power of parallelism that comes with a P2P system may compensate limitations of the technology. For instance, with many machines at our disposal, we can run in parallel more costly tuning algorithms that can meet some (or hopefully all) of the requirements that we have outlined.

7 Data Activation for Files

One of our basic assumptions is that a significant portion of the available data resides in file systems that do little effort to optimize their access or their updates. Still, we envision that the data ring should provide efficient declarative query capabilities over such sources, by means of *data self-activation*.

To illustrate this idea, let us consider a music catalog that has been published as a file in the ring and is queried heavily. Again, we assume that the catalog is presented as an XML view to the users, and queries are translated to a suitable file algebra. Since that physical organization cannot be changed, the data ring may decide, based on self-statistics, to reorganize the catalog in redundant physical structures that are more efficient to access. For instance, it may decide to replicate the catalog (or some part thereof) on a different peer. Another option is to replicate the catalog in a different physical system, e.g. an indexed relational database, in order to answer efficiently queries that access some specific attributes.

Clearly, data self-activation is closely linked to the previously described goals of autonomic administration and self-tuning in particular. For instance, the creation of redundant file indices can be seen as a form of physical design tuning. We choose to emphasize data self-activation as a separate point in the data ring design because scientific data management relies heavily on file systems, and we believe that special attention should be given to the idea of in-situ query processing for data stored in files.

8 Conclusion

The starting point of the data ring project is the observation that large communities of users will increasingly share content over the Internet. We propose the data ring as the means to bring the great benefits of database technology to these communities and to enable them to easily create, administer, and exploit shared content. We have outlined the general principles behind data rings, and discussed some research challenges met on the way to the realization of this vision.

The realization of data rings will clearly require the collaboration of numerous researchers, and we hope to entice other researchers to join us in this endeavor. Indeed, the ideas presented in this paper already lead to a joint project between several research groups, also called Dataring, and supported by the French Agence Nationale de Recherche.

References

[1] S. Abiteboul. Managing an XML Warehouse in a P2P Context. In *CAiSE*, pages 4–13, 2003.

[2] S. Abiteboul, I. Manolescu, and E. Taropa. A Framework for Distributed XML Data Management. In *10th International Conference on Extending Database Technology*, pages 1049–1058, 2006.

[3] S. Abiteboul, Ioana Manolescu, Neoklis Polyzotis, Nicoleta Preda, and Chong Sun. XML processing in DHT networks. In *Proceedings of IEEE ICDE 2008*, pages 606–615, 2008.

[4] S. Abiteboul and N. Polyzotis. The Data Ring: Community Content Sharing. In *Conference on Innovative Data Systems Research (CIDR)*, 2007.

[5] S. Agrawal, S. Chaudhuri, and V.Narasayya. Automated Selection of Materialized Views and Indexes for SQL Databases. In *Proceedings of the 26th Intl. Conf. on Very Large Data Bases*, pages 496–505, 2000.

[6] ActiveXML Web Site. `http://activexml.net`.

[7] Nicolas Bruno, Nick Koudas, and Divesh Srivastava. Holistic Twig Joins: Optimal XML Pattern Matching. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 310–321, 2002.

[8] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. (Available from `http://www.w3.org/TR/wsdl`), 2001.

[9] B. F. Cooper and H. Garcia-Molina. SIL: Modeling and Measuring Scalable Peer-to-Peer Search Networks. In *DBISP2P*, pages 2–16, 2003.

[10] B. F. Cooper and H. Garcia-Molina. Ad Hoc, Self-Supervising Peer-to-Peer Search Networks. *ACM Trans. Inf. Syst.*, 23(2):169–200, 2005.

[11] M. J. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Rec.*, 34(4):27–33, 2005.

[12] L. Galanis, Y. Wang, S. R. Jeffery, and D. J. DeWitt. Locating Data Sources in Large Distributed Systems. In *Proceedings of the 29th Intl. Conf. on Very Large Data Bases*, 2003.

[13] P. Ganesan, Q. Sun, and H. Garcia-Molina. Adlib: A Self-Tuning Index for Dynamic Peer-to-Peer Systems. In *Proceedings of the 21st Intl. Conf. on Data Engineering*, pages 256–257. IEEE Computer Society, 2005.

[14] L.M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang. Optimizing Queries Across Diverse Data Sources. In *Proceedings of the 23rd Intl. Conf. on Very Large Data Bases*, pages 276–285, 1997.

[15] R. Huebsch, J.M. Hellerstein, N. Lanham, B.T. Loo, S. Shenker, and I. Stoica. Querying the internet with PIER. In *Proceedings of 19th International Conference on Very Large Databases*, 2003.

[16] V. Josifovski, P. Schwarz, L.M. Haas, and E. Lin. Garlic: a new flavor of federated query processing for DB2. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 524–532, 2002.

[17] P. Krishna and T. Johnson. Index replication in a distributed B-tree. In *COMAD*, 1994.

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, 2001.

[19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, 2001.

[20] I. Tatarinov, Z. G. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, X. Dong, Y. Kadiyska, G. Miklau, and P. Mork. The piazza peer data management project. *SIGMOD Rec.*, 32(3):47–52, 2003.