XML DATA INTEGRATION IN PEER-TO-PEER DATA MANAGEMENT SYSTEMS

Tadeusz Pankowski
Institute of Control and Information Engineering, Poznań University of Technology, Poland
Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland

Keywords: XML data integration, query propagation, XML functional dependencies, P2P data management.

Abstract: P2P systems are commonly accepted as an efficient means of sharing data among large, diverse and dynamic set of users. Nowadays sharing data imposes new challenges in P2P systems concerning supporting advanced querying beyond simple keyword-based retrieval. We assume that each peer stores schema of its local data, mappings to some other peers, and schema constraints (functional dependencies). The goal of the integration is to answer queries formulated against arbitrarily chosen peers. The answer consists of data stored in the queried peer as well as data of its direct and indirect acquaintances. We focus on the problem of query propagation and merging partial answers in such environment. We show how XML functional dependencies defined over schemas, determine the selection of the merging mode of partial answers to increase information content of the answer by recovering some missing values. We show how the discussed method has been implemented in SixP2P system (*Semantic Integration of XML data in P2P environment*).

1 INTRODUCTION

Peer-to-peer (P2P) data management systems are becoming increasingly attractive as an efficient means of sharing data among large, diverse and dynamic sets of users (Madhavan and Halevy, 2003; Tatarinov and Halevy, 2004). In such setting, the autonomous computing nodes (the *peers*) cooperate to share resources and services. The peers are connected to some other peers they know or discover (Bernstein et al., 2002; Koloniari and Pitoura, 2005; Pankowski, 2006). In such systems, the user issues queries against an arbitrarily chosen peer and expects that the answer will include relevant data stored in all P2P connected data sources. The data sources are related by means of schema mappings, which are used to specify how data structured under one schema (the source schema) can be transformed into data structured under another schema (the target schema) (Fagin et al., 2004; Fuxman et al., 2006). A query must be propagated to all peers in the system along semantic paths of mappings and reformulated accordingly. The partial answers must be merged and sent back to the user peer (Melnik et al., 2005; Yu and Popa, 2004).

In this paper, we focus on the impact of the relationship between schema constraints and queries on the way of query execution (query propagation and merging answers delivered by interrogated peers). We show how some missing values (denoted by null) may be inferred (discovered) in the integration process. In particular, in Proposition 2.1 we formulate a formal condition saying when it is reasonable to use so called *full merge* while merging partial answers. The discussed methods were implemented in SixP2P system. The system is based on formal foundations underlying this paper, and implements algorithms translating high-level specifications of schemas, constraints and queries into XQuery programs performing data transformation, query evaluation and discovering missing data (Brzykcy et al., 2007).

Section 2 introduces a running example and gives motivation of the research. We discuss query execution strategies and show how the result of queries depends on the chosen strategy. In Section 3 we discuss implementation of SixP2P system. We sketch its architecture and illustrate the way the queries and answers are propagated in the system. Section 4 concludes the paper.

Pankowski T. (2008).
XML DATA INTEGRATION IN PEER-TO-PEER DATA MANAGEMENT SYSTEMS.
In *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, pages 296-300
DOI: 10.5220/0001529602960300

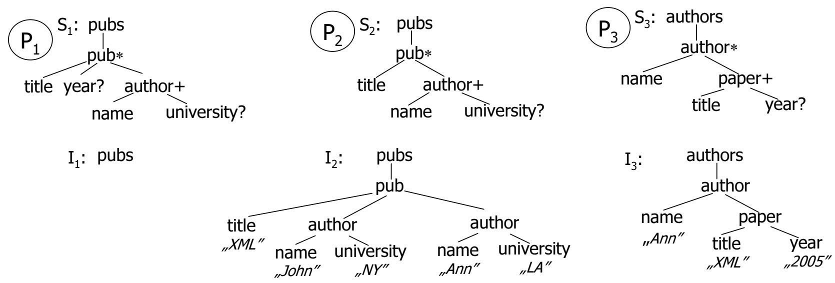

Figure 1: XML schema trees S_1, S_2, S_3, and their instances I_1, I_2 and I_3, located in peers P_1, P_2, and P_3.

2 QUERY EXECUTION STRATEGIES

In Figure 1 there are three peers P_1, P_2, and P_3 along with XML schema trees, S_1, S_2, S_3, and schema instances I_1, I_2, and I_3, respectively. Further on, we will assume that XML schemas can be represented by tree-pattern formulas (Arenas and Libkin, 2005; Pankowski et al., 2007).

In P2P data integration systems a query formulated against an arbitrary target schema (owned by a target peer) must be propagated to all partners of the target peer, these peers propagate it further to their partners, etc. In this way the query can reach all sources, which can contribute to the final answer. Partial answers are merged step-by-step and successively sent towards the target peer. In such scenario the following three issues are of special importance:

1. *Query propagation* – using the information provided by the query and by available schemas, the peer has to decide who to send (propagate) the query to, and whether a coming propagation should be accepted in order to avoid cycles and to increase the expected amount of information.
2. *Query reformulation* – a query received and accepted by P_i from P_j has to be reformulated in such a way that it can be evaluated over S_i and its answer conforms to S_j.
3. *Merging partial answers*. A peer can decide whether the received answers should be merged with or without the whole peer's local instance. This decision is made based on the functional dependencies defined over the local schema.

We assume that a peer makes decision locally based on its knowledge about its schema and schema constraints and about the query that should be executed and propagated. The chosen strategy and the way of merging partial answers determine both the final answer and the cost of the execution.

We will use XML functional dependencies (XFDs) (Arenas, 2006) as schema constraints. Over S_3 the following XFD can be defined:

$$\begin{array}{l} /authors/author/paper/title \rightarrow \\ /authors/author/paper/year \end{array} \quad (1)$$

This XFD can be specified as the formula:

$$/authors/author/paper[title = x_{title}]/year = x_{year}$$

meaning that each value of x_{title} uniquely determines the text value x_{year} of *year*.

Let us consider some possible strategies of execution query q over S_1

$$\begin{array}{l} q := /pubs[pub[title = x_{title} \wedge year = x_{year} \\ \quad \wedge author[name = x_{name} \\ \quad \wedge university = x_{univ}]]] \wedge x_{name} = \text{"}John\text{"}, \end{array}$$

where the first conjunct is the schema, variables $x_{title}, x_{year}, x_{name}$, and x_{univ} are bound to text values of an instance of S_1; $x_{name} =$ "*John*" is the query qualifier. The answer should contain information stored in all three sources shown in Figure 1.

Thus, one of three strategies can be realized:

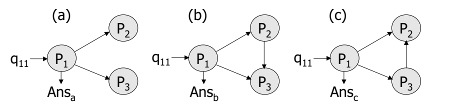

Figure 2: Three execution strategies of the query q.

Strategy (a). Query q is sent to P_2 and P_3, where it is reformulated to, respectively, q_{21} (from P_2 to P_1) and q_{31} (from P_3 to P_1). The answers $q_{21}(I_2)$ and $q_{31}(I_3)$ are returned to P_1. In P_1 these partial answers are merged with the local answer $q_{11}(I_1)$ and a final answer Ans_a is obtained. This process can be written as follows ($\sqcup$ denotes the merge operation):

$$\begin{array}{lcl} Ans_a & = & \sqcup\{Ans^a_{11}, Ans^a_{21}, Ans^a_{31}\}, \\ Ans^a_{11} & = & q_{11}(I_1) = \{(x_{title} : \bot, x_{year} : \bot, x_{name} : \bot, \\ & & \quad x_{univ} : \bot)\}, \\ Ans^a_{21} & = & q_{21}(I_2) = \{(x_{title} : XML, x_{name} : John, \\ & & \quad x_{univ} : NY)\}, \\ Ans^a_{31} & = & q_{31}(I_3) = \{(x_{name} : \bot, x_{title} : \bot, x_{year} : \bot)\}, \\ Ans_a & = & \{(x_{title} : XML, x_{year} : \bot, x_{name} : John, \\ & & \quad x_{univ} : NY)\}. \end{array}$$

Strategy (b). It differs from strategy (a) in that P_2 after receiving the query propagates it to P_3 and waits for the answer $q_{32}(I_3)$. The result is equal to Ans_a:

$$\begin{aligned} Ans_b &= \sqcup\{Ans^b_{11}, Ans^b_{21}, Ans^b_{31}\} = \\ &= \{(x_{title}:XML, x_{year}:\bot, x_{name}:John, \\ &\quad x_{univ}:NY)\}, \end{aligned}$$

Strategy (c). In contrast to the strategy (b), the peer P_3 propagates the query to P_2 and waits for the answer. Next, the peer P_3 decides to merge the obtained answer $q_{23}(I_2)$ with the whole its instance I_3. The decision is based on the existence of the functional dependency (1) and Proposition 2.1.

$$\begin{aligned} Ans_c &= \sqcup\{Ans^c_{11}, Ans^c_{21}, Ans^c_{31}\}), \\ Ans^c_{23} &= q_{23}(I_2) = \{(x_{title}:XML, x_{year}:\bot, \\ &\quad x_{name}:John)\}, \\ Ans^c_{31} &= q_{31}(\sqcup\{I_3, Ans^c_{23}\}) = \\ &= \{(x_{title}:XML, x_{year}:2005, x_{name}:John)\} \\ Ans_c &= \{(x_{title}:XML, x_{year}:2005, x_{name}:John, \\ &\quad x_{univ}:NY)\}. \end{aligned}$$

While computing the merge $\sqcup\{I_3, Ans^c_{23}\}$ a missing value of x_{year} is discovered. Thus, the answer Ans_c provides more information than Ans_a and Ans_b.

The above example shows that it is important to decide which of two merging modes should be used in the peer while partial answers are to be merged:

- *partial merge* – all partial answers are merged without taking into account the source instance stored in the peer (e.g. the strategy (b));
- *full merge* – the whole source instance in the peer is merged with all received partial answers; during this operation XFDs are used to discover missing values; finally the query is evaluated on the result of the merge (e.g. the strategy (c)).

Criterion of the selection is the possibility of discovering missing values during the process of merging. To make the decision one has to analyze XFD constraints specified for the peer's schema and the query qualifier.

Proposition 2.1 states the condition when there is no sense in applying full merge because no missing value can be discovered (Pankowski, 2008).

Proposition 2.1. *Let $S(\mathbf{x})$ be a schema, q be a query with qualifier $\psi(\mathbf{y})$, $\mathbf{y} \subseteq \mathbf{x}$, and I_A be an answer to q received from a propagation. Let $\psi(\mathbf{z}) = x$ be an XFD defined over $S(\mathbf{x})$. If one of the following two conditions holds: (a) $x \in \mathbf{y}$, or (b) $\mathbf{z} \subseteq \mathbf{y}$, then no missing value can be discovered by full merge, i.e.*

$$q(merge(I, I_A)) = merge(q(I), I_A).$$

3 DATA INTEGRATION IN SIXP2P

The discussed method of semantic data integration is realized in the SixP2P system. The overall architecture of the system is in Figure 3, and the software structure is given in Figure 4.

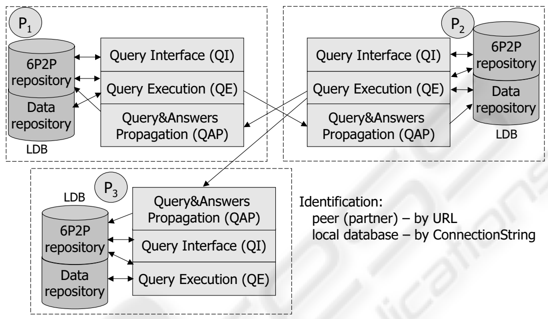

Figure 3: Overall architecture of SixP2P.

Each peer in SixP2P has its own local database consisting of two parts: data repository of data available to other peers, and 6P2P repository of data necessary for performing integration processes (information about partners, schema mappings, schemas, constraints, partial answers, etc.). Using the query interface (QI) a user formulates a query. The query execution module (QE) controls the process of query reformulation, query propagation to partners, merging of partial answers, discovering missing values, and returning partial answers (Figure 5). Communication between peers (QAP) is realized by means of Web Services technology. Layers in Figure 4 show tasks realized by particular modules.

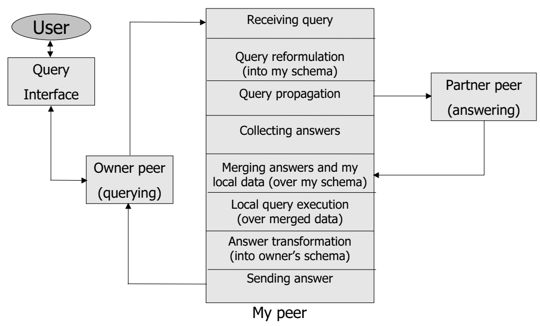

Figure 4: Software architecture of SixP2P.

In Figure 5 there is the (simplified) structure of 6P2P repository showing the propagation of queries and answers in the SixP2P system consisting of three peers: P_1, P_2, and P_3. Specification of the query is translated into executable form to *myQuery* (an XQuery program to be executed over the local database) and to *tgtQuery* (an XQuery program transforming the obtained answer into the target schema). The query q_1 is propagated to (all or some) partners of P_1 – among them also to P_1 itself. Each propagation is recorded in table *Propagations*, where: *propagID* identifies the propagation; *qryPosId* identifies the position in table *Queris*; *srcPeer* is the URL of the source partner, where the query has been propagated; *srcAnswer* is the answer obtained from the *srcPeer*.

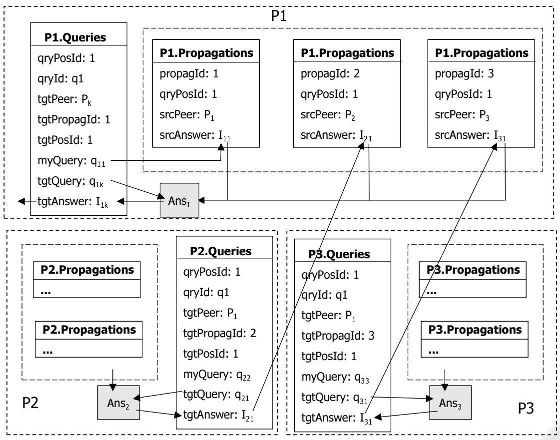

Figure 5: Query and answers propagation in SixP2P.

All *srcAnswer*s are merged (using full or partial mode) resulting to the Ans_1. Next, *tgtQuery* is evaluated over Ans_1 to obtain *tgtAnswer*, which is ultimately sent to *tgtPeer* and stored in *tgtPeer*'s *Propagations* table in the tuple identified by the pair $(tgtPropagId, tgtPosId)$. The evaluation removes duplicates and considers key constraints.

4 CONCLUSIONS

The paper presents a novel method for schema mapping and query reformulation in XML data integration systems in P2P environment. We discussed some issues concerning query propagation strategies and merging modes, when missing data is to be discovered in the P2P integration processes. We showed, how to use functional dependencies to select the way of query propagation and data merging, to increase the information content of the answer. The approach is fully implemented in SixP2P system. We present its general architecture, and sketched the way how queries and answers are sent across the P2P environment. In SixP2P, schemas, schema constraints, schema mappings, and queries are specified in a uniform and precise way. We develop algorithms for automatic generation of XQuery programs which perform operations of query reformulation and data merging.

ACKNOWLEDGEMENTS

The work was supported in part by the Polish Ministry of Science and Higher Education under Grant N516 015 31/1553.

REFERENCES

Arenas, M. (2006). Normalization theory for XML. *SIGMOD Record*, 35(4):57–64.

Arenas, M. and Libkin, L. (2005). XML Data Exchange: Consistency and Query Answering. In *PODS Conference*, pages 13–24.

Bernstein, P. A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., and Zaihrayeu, I. (2002). Data management for peer-to-peer computing : A vision. In *WebDB*, pages 89–94.

Brzykcy, G., Bartoszek, J., and Pankowski, T. (2007). Semantic Data Integration in P2P Environment using Schema Mappings and Agent Technology, *AMSTA 2007*. In *Lecture Notes in Computer Science 4496*, pages 385–394. Springer.

Fagin, R., Kolaitis, P. G., Popa, L., and Tan, W. C. (2004). Composing schema mappings: Second-order dependencies to the rescue. In *PODS*, pages 83–94.

Fuxman, A., Kolaitis, P. G., Miller, R. J., and Tan, W. C. (2006). Peer data exchange. *ACM Trans. Database Syst.*, 31(4):1454–1498.

Koloniari, G. and Pitoura, E. (2005). Peer-to-peer management of XML data: issues and research challenges. *SIGMOD Record*, 34(2):6–17.

Madhavan, J. and Halevy, A. Y. (2003). Composing mappings among data sources. In *VLDB*, pages 572–583.

Melnik, S., Bernstein, P. A., Halevy, A. Y., and Rahm, E. (2005). Supporting executable mappings in model management. In *SIGMOD Conference*, pages 167–178.

Pankowski, T. (2006). Management of executable schema mappings for XML data exchange. In *Database Technologies for Handling XML Information on the Web, EDBT 2006 Workshops*, Lecture Notes in Computer Science 4254, pages 264–277.

Pankowski, T. (2008). Pattern based XML data integration in P2P environment. *submitted*.

Pankowski, T., Cybulka, J., and Meissner, A. (2007). Reasoning About XML Schema Mappings in the Presence of Key Constraints and Value Dependencies. In *Web Reasoning and Rule Systems*, Lecture Notes in Computer Science 4524, pages 374–376.

Tatarinov, I. and Halevy, A. Y. (2004). Efficient query reformulation in peer-data management systems. In *SIGMOD Conference*, pages 539–550.

Yu, C. and Popa, L. (2004). Constraint-Based XML Query Rewriting For Data Integration. In *SIGMOD Conference*, pages 371–382.