On Updating in XML Peer-to-Peer Databases

Adam Šenk, Michal Valenta

Czech Technical University in Prague, Faculty of Information Technology
Thákurova 9, Prague, Czech Republic
{senkadam,valenta}@fit.cvut.cz

Abstract. Distributed databases are increasingly used in many applications and software systems. Peer-to-Peer databases are interesting part of distributed databases field. This type of distributed database management system allows to connect stand-alone databases via a computer network. Connected databases cooperate on processing of their tasks with other connected databases.
The peer-to-peer databases contains a lot of various distributed and replicated data in various forms. The correct data coordination is the crucial in peer-to-peer databases, it is necessary for correct processing of queries and other database requests. Our work is focused on updates in XML peer-to-peer databases. We provide a review of a peer-to-peer database management system architecture and problems related to implementation of updates. Finally, we consider the requirements on semantic mapping, the part of peer architecture responsible for correct reformulation of global query or update.

Keywords: Peer-to-Peer, XML, Update, Semantic mapping

1 Introduction

Last year, the XML (Extensible Markup Language) celebrated his fifteenth birthday and it has become one of the leading data format. Most of the data published on Web are represented by the XML. The XML is also popular in a data exchange between various systems. XML-native databases use XML as a data format for persistent data storing. Generally speaking, the XML is very popular and variable format and it is used in many systems. Nowadays, most of these systems, that use an XML format, can be seen as big distributed database systems with some limitations. They have many properties that are similar to peer-to-peer database systems properties.

Peer-to-peer database management system (DBMS) is a distributed DBMS with some major differences. It is composed of a large number of database nodes with various bindings. The structure of such system is complicated, changeable and cannot be clearly described by a global schema of distribution. Processing of queries, updates, transactions and other database operations in peer-to-peer DBMS brings new challenges for database developers as they are facing problems caused by the architecture of such systems.

V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2013, pp. 139–148, ISBN 978-80-248-2968-5.

In our work, we investigate an update processing in XML peer-to-peer systems. The problem is that the data stored in the peer-to-peer DBMS are very often interrelated. So it means that updates done on this data could break some dependency or logical structure of stored documents. It is necessary to propagate changes done in one database node to other nodes. We concentrate on basic problem related to update processing in XML peer-to-peer DBMS - identification of data relations in peer-to-peer systems. We analysis what are the biggest problems related to identification of entities, their relations and to coordination of related data updates.

The structure of this paper is the following: Section 2 is summary of related works and known scientific results from field of peer-to-peer DBMS. In Section 3 we define peer-to-peer DBMS and describe architecture and properties of peer-to-peer DBMS. In Section 4 we describe problems related to update processing and to identification of data relations. Section 5 we analyse the requirements on important part of peer-to-peer architecture - the semantic mapping. In Section 6 we discuss the contribution of this work and also the future work of our research.

2 Related Work

There are many woks related to a peer-to-peer databases topic. The book [11] provides a great theoretical background. It reviews the whole distributed databases topic and peer-to-peer databases are described in this context.

A lot of research was done on a field of relational peer-to-peer databases. There exist many distributed peer-to-peer database management systems, one of them is coDB [6]. Greco and Scarcello [7] focused on inconsistencies in query results realized in peer-to-peer databases. Some works are also focused on updates in peer-to-peer databases. Datta [5] and colleagues propose the push/pull strategy for an update processing. Kantere and colleagues introduces distributed triggers for an update processing in relational peer-to-peer databases in [9].

Angela Bonifati and colleagues focused on XML peer-to-peer databases in [2]. They provide a solution for a query reformulation in heterogeneous XML schemas including data and meta-data conflicts. This solution is based on assumption, that each peer knows the structure of data stored in a neighbour peer. Unfortunately, they did not solve updates.

3 Peer-to-Peer Database Managements System

The distributed database management system is a several data processing systems (not necessarily homogeneous) connected via computer network. They communicate and cooperate on data processing. Özsu and Valduriez [11] define a distributed database as a collection of multiple, logically interrelated databases distributed over a computer network. A distributed database management system (DDBMS) is then defined as the software system that permits the management of the distributed database and makes the distribution transparent to the users.

Peer-to-Peer DBMS is distributed system that is composed of stand-alone databases and their bindings. It can be represented by a set of nodes and edges (directed or undirected). Nodes represent some stand-alone DBMS system (SQL database, NoSQL database, distributed database ...) and edges represent bindings between these nodes. The example of XML peer-to-peer system is on Figure 1.

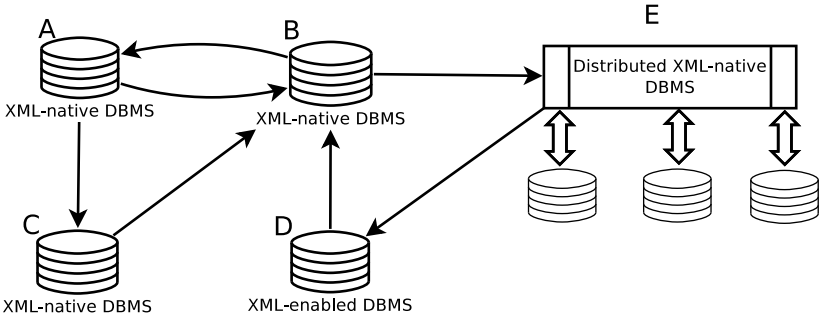

Fig. 1. Example of a schema of some peer-to-peer XML database

There are three significant properties that differ peer-to-peer DBMS from other DDBMS. The first difference is a big amount of nodes in peer-to-peer DBMS. Classical DDBMS with global schema of distribution consist of tens of nodes. Peer-to-Peer DBMS can consists from hundreds or thousands nodes. It causes a massive distribution of data.

The second significant property is a big heterogeneity of such systems and a big autonomy of nodes. Peer-to-Peer DBMS can consists from various database nodes with very various data and data representations.

The autonomy of nodes is the third significant property. The nodes can connect or disconnect any time. It makes the structure of peer-to-peer DBMS very changeable.

3.1 Architecture of Peers

The basic requirements on each DDBMS are as follows:

- **Query processing** - the DBMS must be able to discover all relevant data distributed over the database and efficiently execute the query.
- **Data integration** - the DBMS must be able to discover all related data, even if their representation or schema is different.
- **Data consistency** - the DBMS must maintain the consistency between duplicated data in distributed database.

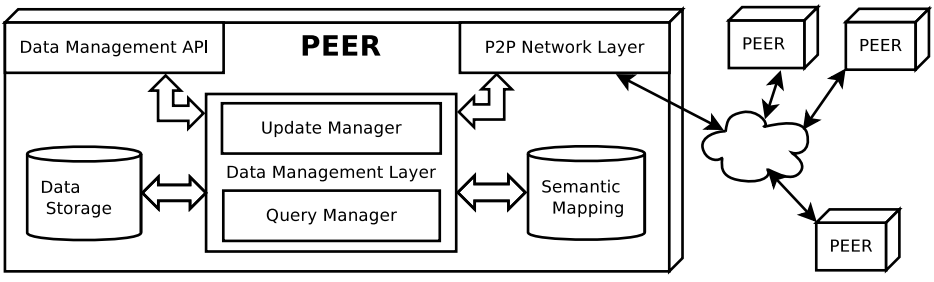

Fig. 2. Peer Architecture

Each peer in a peer-to-peer system must be able to locate and access data distributed and stored in other peers. In Figure 2 is the architecture of a single peer that is to meet this requirements. The architecture consist of four corner stones.

Data management layer is responsible for a query processing. It can have many sub-parts, e.g. the query manager, the update manager and the cache manager. Queries can ask for local data stored in asked peer or for global data distributed in the whole database. Data management layer is also responsible for an execution of remote request send by some other peer.

Data management API is an interface for peer users. The two most common parts are a client containing graphic or text user interface and libraries and public methods for developers. The client allows to submit queries and other requests. It should also provide tools for database tuning and setting its properties. Developers can use application programmable interface to integrate peer functionality into an application.

P2P Network Sublayer is an interface for communication with connected peers over the computer network. It receives requests from the data management layer and distributes them in the whole peer-to-peer DBMS. It also receives messages from other peers and send them to the data management layer to be proceeded.

Persistent storage consists of two basic repositories - *data storage* repository that contains local data and *semantic mapping* that contains meta-information needed for remote query processing. Data management layer uses meta-information for query reformulation when the data stored in the data storage are queried by a remote peer.

Theoretically, the data repository can be replaced by the classic DBMS. Then we see all other layers as a system allowing the connection of such DBMS into a distributed peer-to-peer database.

4 Propagation of Updates

The update in peer-to-peer DBMS is an operation that persistently changes the data stored in some node. The problem of this operation is how to propagate these changes to all nodes in peer-to-peer network. The data in a peer-to-peer database can be replicated in many nodes. It is necessary to distribute the changes over the whole system network. Otherwise, the database consistency will be broken.

To propagate the updates correctly, all interrelated data stored in a peer-to-peer database must be found. The identification of entities, their relations and their states in semi-structured data like XML is a complex problem. We state major problems related to propagation of updates that we are facing.

4.1 Entities in XML

In general, an entity is an existing or real thing. Entity in context of databases is a thing which data can be stored.

The XML is an ordered tree structure consist of edges and nodes. There are three types of node: *element*, *attribute* and *text*. The hierarchical structure of the XML and the three types of nodes in XML tree bring complications to an entity recognition.

An example of two different XML representation of one entity is given in Figures 1.1 and 1.2. As you can see, the structure of an XML fragment is absolutely different in these two cases, but it is obvious that both examples represent the same entity.

Listing 1.1. Book and Authors version 1

```
<book isbn="0-079-13702-4" name="XML Complete" price="49.99
  <author id="1001" firstname="Steven" surname="Holzner" />
<book>
```

Listing 1.2. Book and Authors version 2

```
<book>
  <isbn>0-079-13702-4</isbn>
  <name>XML Complete</name>
  <price>49.99<price>
  <author>
    <id>1001</id>
    <firstname>Steven</firstname>
```

```
    <surname>Holzner</surname>
  </author>
<book>
```

Equality of XML nodes is defined in [8].

Definition 1. *Let name(x), val(x), att(x) and ele(x) be a functions. Function name(x) returns name of given node, function val(x) returns value of given attribute or text node, function att(x) returns list of attribute of given element node and function ele(x) returns the list of descendants of given element node. Two nodes u and v are equal (u = v) if the following conditions are satisfied:*

1. $name(u) = name(v)$
2. *if* u, v *are attribute or text nodes, then* $val(u) = val(v)$
3. *if* u, v *are element nodes, then:*
   (a) *if* $att(u) = a_1, ..., a_m$, *then* $att(v) = a'_1, ..., a'_m$ *and there is a permutation* π *on* $1, ..., m$ *such that* $val(a_i) = val(a_{\pi(i)})$ *for* $i = 1, ..., m$
   (b) *if* $ele(u) = [u_1, ..., u_k]$, *then* $ele(v) = [v_1, ..., v_k]$ *and* $val(u_i) = val(v_i)$ *for* $i = 1, ..., k$

Equality of Entities is not the same as the equality of XML nodes. Two XML nodes can represent one entity even if they are not equal (see Listing 1.1 and Listing 1.2). We define the **entity equality** as follows:

Definition 2. *Two XML nodes u and v are entity equal* ($u =_{ee} v$) *iff they represent the same entity.*

4.2 State of Entity

Each entity can have different states. State is a part of data stored in entity that can differ node by node in one peer-to-peer database. In Listing 1.3 the example is given. This XML code represents the same entity as in two previous example (Listing 1.1 and Listing 1.2), but the value of *price* attribute differs.

Listing 1.3. Book and Authors version 3

```
<book isbn="0-079-13702-4" name="XML Complete" price="56.99
  <author>
    <id>1001</id>
    <firstname>Steven</firstname>
    <surname>Holzner</surname>
  </author>
<book>
```

Attribute *price* is state attribute of an entity and can be different in different nodes. The updates done on parts of XML that represent some state are local and they cannot be distributed over peer-to-peer database.

4.3 Primary Key

The *primary key* is a unique identifier of a data entity. It is the minimal part of an entity which determines its uniqueness. The primary key is used by the data management layer for identifying the updated entity. With primary key defined, it is much more easier to find related data in a distributed peer-to-peer database.

Definition 3. *Lets have two entities* e_1, e_2 *and a primary key function* $PK(e)$ *which returns a primary key and its value for given entity. Then* $PK(e1) = PK(e2) \Leftrightarrow e_1 = e_2$.

The definition 4 of the primary key for relational databases proposed in [4] became a standard for SQL databases. This definition is not usable for a semi-structured hierarchical XML data and it has to be adapted.

Definition 4. *One domain (or combination of domains) of a given relation has values which uniquely identify each element (n-tuple) of that relation. Such a domain (or combination) is called a primary key. In the context of XML hierarchical structure the problem of keys become much more complicated.*

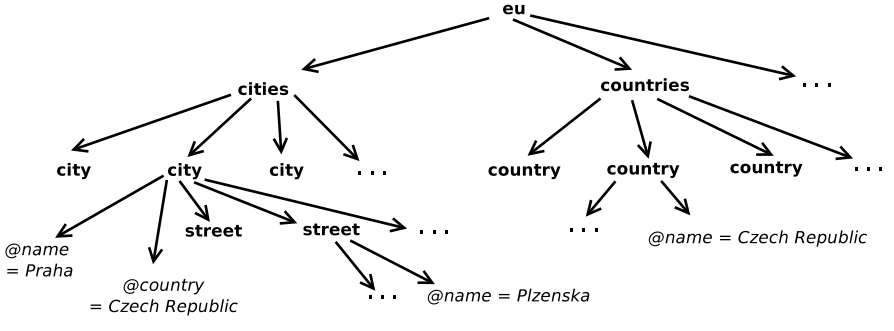

Fig. 3. eu.xml - example of XML data

Let us introduce an example of an XML document in Figure 3 called *eu.xml*. We want to demonstrate problems caused by the hierarchical structure of XML related to the primary key. The *eu.xml* document stores data of member countries in European Union. As you can see, there are three elements with *@name* attribute - **country**,**city** and **street**. The *@name* attribute of **country** element should be unique (there are not two countries with the same name in European Union and there are not two countries with the same name in whole world either). Is the *@name* attribute unique for the **city** element too? Definitely not. There are many examples of cities with the same name (In the Czech Republic, there is one other city that has the same name as the capital city - Prague). There are also examples of cities having the same name as the country (Luxembourg, the member state of European Union has capital city called Luxembourg).

The previous example demonstrates that in an XML data it is necessary to define not only the primary key but also its context. The attribute *@name* is definitely a primary key, but only in context of **countries** elements. That is the reason why keys in XML documents are defined relatively. The definition of *relative keys* was proposed in [3]. We use the XPath language [1] for a key context definition.

Definition 5. *Let P_c and P_t be an XPath expression and let S be a set of XPath expressions. The primary key is defined as $\varphi = P_c(P_t(S))$. If for any node v reached via P_c and for any nodes v_1, v_2 reachable from v via P_t the sets of values $s_1 and s_2$ reached from v_1, v_2 via S satisfies: $s_1 \neq s_2$, then XML document D satisfies φ.*

Example 1. Primary key φ_1 says that *eu.xml* document in Figure 3 is valid if any *@name* attribute of **country** element is unique in a context of **countries** element:

$$\varphi_1 = /eu/countries(/country(@name))$$

5 Semantic Mapping for XML Updates

In chapter 3, we described the architecture of peer-to-peer DBMS. Then we described, in chapter 4, the three major problems that we are facing when we implements updates in XML peer-to-peer DBMS. The analysis done in previous chapters shows the requirements for semantic mapping:

- **Entity recognition** - The DBMS must be able to recognize a single data entity in the hierarchical XML model.
- **Correct update propagation** - The DBMS must to recognize which updates must be propagated in other peers and which not.
- **Common format** - The DBMS must transform XML data to one common format. The data stored in XML can have various formats, so it is necessary to define format that enables the recognition of similarities.

5.1 Definition of Keys

Semantic mapping extracts all important data and provides them to the data management layer. In Section 4.3 we defined the primary key in XML. It is obvious that the semantic mapping must define primary keys of XML data stored in data storage. Primary keys make identification of equal entity easier. Without primary keys is almost impossible to process queries and updates on such complex and heterogeneous data that can be found in peer-to-peer databases.

5.2 Global and Local data

The correct update propagation is one of the requirements on semantic mapping that we state in this section. The data that must be propagated in other peers are called *global* data. The data that are not replicated and are valid in context of single peer are called *local* data. The data management layer cannot automatically recognize the local and global data, this information must be stored in the semantic mapping.

5.3 General model

The most important and also difficult function of the semantic mapping is to map all XML documents to one general model. This model ignores the differences in structure of XML data and allow to compare entities stored in XML format. Developing of such model and developing of mapping on this model is non-trivial task.

6 Conclusion and Future work

In our paper we described the peer-to-peer database management systems. We provided the review of the concept and the description of architecture of the basic unit - peer. Then we concentrated on updates in XML peer-to-peer databases and state the problems to solve when such system is implemented. Finally we state the requirements on the semantic mapping that must be fulfilled in the peer-to-peer database to keep the stored data consistent after updating. Although we do not provide any details we believe that analysis done in our work is a good base for our future research.

Our future work will focus on developing of peer-to-peer DBMS which will allow to connect various independent XML storages in one distributed database. The summarize of information in this paper gives us excellent background. The aim of our future work is to solve processing of updates and other events in distributed system.

References

1. XML Path Language (2013), http://www.w3.org/TR/xpath/
2. Bonifati, A., Chang, E., Ho, T., Lakshmanan, L.V., Pottinger, R., Chung, Y.: Schema mapping and query translation in heterogeneous p2p xml databases. The VLDB Journal 19(2), 231–256 (Apr 2010), http://dx.doi.org/10.1007/s00778-009-0159-9
3. Buneman, P., Davidson, S., Fan, W., Hara, C., Tan, W.C.: Keys for xml. Computer Networks 39(5), 473 – 487 (2002), http://www.sciencedirect.com/science/article/pii/S1389128602002232
4. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13(6), 377–387 (Jun 1970), http://doi.acm.org/10.1145/362384.362685

5. Datta, A., Hauswirth, M., Aberer, K.: Updates in highly unreliable, replicated peer-to-peer systems. In: Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on. pp. 76 – 85 (may 2003)
6. Franconi, E., Kuper, G., Lopatenko, A., Zaihrayeu, I.: Queries and updates in the codb peer to peer database system. In: Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. pp. 1277–1280. VLDB '04, VLDB Endowment (2004), http://dl.acm.org/citation.cfm?id=1316689.1316813
7. Greco, G., Scarcello, F.: On the complexity of computing peer agreements for consistent query answering in peer-to-peer data integration systems. In: Proceedings of the 14th ACM international conference on Information and knowledge management. pp. 36–43. CIKM '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1099554.1099564
8. Hartmann, S., Link, S.: Efficient reasoning about a robust xml key fragment. ACM Trans. Database Syst. 34(2), 10:1–10:33 (Jul 2009), http://doi.acm.org/10.1145/1538909.1538912
9. Kantere, V., Manoubi, M., Kiringa, I., Sellis, T., Mylopoulos, J.: Peer coordination through distributed triggers. Proc. VLDB Endow. 3(1-2), 1561–1564 (Sep 2010), http://dl.acm.org/citation.cfm?id=1920841.1921038
10. Masud, M., Kiringa, I.: Transaction processing in a peer to peer database network. Data & Knowledge Engineering 70(4), 307 – 334 (2011), http://www.sciencedirect.com/science/article/pii/S0169023X10001527
11. Özsu, M., Valduriez, P.: Principles of distributed database systems. Springer (2011)