Models XP for Rewriting XPath Queries

Nicoleta Liviana Tudor

Department of Computer Science, Petroleum-Gas University of Ploiesti,
39, Bucureşti Street, 100680, Romania,
LTudor@upg-ploiesti.ro

Abstract: This paper defines XP models for classes of XPath queries stored in cache, as materialized views. After declaration of issue of the correspondence between the tree models used for classes of XPath queries stored in cache and the set of trees associated to a XML document, it follows the solutions for rewriting the XPath views by transformation of patterns of trees. Author's personal contribution consists in modelling the set of trees associated to XP queries, for a multitude of constraints of XPath expressions and description of functions of correspondence in XP{ /, //, *, [] } representation. Verification of possibility to return the result of a query using the views materialized in cache requires the analysis of compatibility of tree models associated to XPath queries and XPath views materialized in cache. Finding a morphism of XP models demonstrates the real possibility of rewriting the XP view. This paper describes a method for establishing a semantic cache of XPath views. Composition of queries using a semantic cache of XP views, assumes the existence of a query which, by composition with a view from cache, will return the result of query.

Keywords: XPath query, rewriting views, XP models, XML tree, morphism, composition of queries, cache.

1. Introduction

Data processing as a business object or XML Web services, used within data exchange between applications, requires XML data format [17]. Relational systems convert XML data into a relational format which may be effectively memorized and queried as a set of relations.

Various approaches of storing XML data in relational databases suggest use of metadata, of generic relational table diagrams (pre-defined) including mapping of XML documents. Kossman şi Florescu [1] represented XML documents using graphs, where each edge is a tuple of a relation. This approach uses recursive SQL99 queries including constructions of assessment of XML queries. Zhang, DeWitt and others [2] proposed a system labeling each node by numbers obtained by pre and post order crossing of tree.

To solve multiple joins, Yoshikawa, Amagasa and others [3] suggest storing information about each node of tree in the XML document, especially the path from the root of the tree until each leaf node. The algorithm of translation of XML data as proposed by Yoshikawa and Amagasa is appropriate only for non-recursive data and fails to succeed in correct results when XML data have ancestors with the same label within representation tree [4]. This algorithm requires use of joins with μ operators (<*or*>), implying a very expensive processing [2].

Bohannon, Freire and others [5] have cost based approach using information from XML diagram to select an alternative of execution of XML queries for the lowest price.

The relational diagram allows elaboration of algorithms for translation of XQuery queries into SQL queries, as XQuery is a standard language for XML data query. For instance, Oracle XML [3] allows storing the entire XML document using CLOB type data, the assessment of XML queries being similar to process of queries in native XML databases. Microsoft SQL Server enables creation of relational data XML views. Query of views using XPath language is restricted at XPath expressions.

For optimization of XPath queries, a semantic cache could memorize XML views. In order to avoid repeated connection with backend database, views materialized in cache are subject to queries. This type of middle-tier cache became very popular in Web applications using relational databases [6].

Mandhani and Suciu [7] have suggested a method of creating a semantic cache storing XPath views, used in query processing. Views materialized in cache are XPath expressions and queries could be XQuery or XPath fragments, for whose execution the systems checks firstly in cache if it may return the result. They memorized the views cache in relational tables showing the efficiency of the techniques for selection of views.

Lee and Wesley Chu [8] presented a semantic cache for Web databases relying upon translations of queries and made an analysis of compatibilities of queries with views memorized in cache. The semantic cache proposed by Lee and Wesley Chu consists of a hash table with input data type (key, value), where the key is a semantic description following the queries made and value contains results of the queries associated to the key. In this type of cache, the semantic views use only conjunctive predicates, queries being transformed in conjunctive components.

The paper is organized as follows:

- the section Rewriting XPath views describes the issue of rewriting the XML data based on XPath views, within relational databases
- the section Tree models for XPath queries is dedicated to tree models associated to certain classes of XPath queries, using XP{ /, //, *, [] } representation within context of rewriting XPath views and describes the correspondence functions
- the section Morphism of XP models demonstrates that there is a morphism of XP models on the set of functions of transformation of tree models associated to XP queries
- the section Semantic Cache of XP Views describes the composition of XPath queries and the way of processing a query using a semantic cache of XP views
- the section Conclusions.

2. Rewriting XPath Views

A standard approach for optimization of XPath queries uses rewriting algorithms of XPath view. The problem of data rewriting has been thoroughly studied in the relational model of data [7], [9]. XML data permit also equivalent rewriting using navigation of interior structure of a XPath view.

Certain systems as XPath 2.0 standard, allow XPath queries [10], using identifiers for nodes at execution of intersection of sets of nodes. These nodes' identifiers enable multiple rewriting by intersection of results of views.

Lakshmanan, Gao et al [11, 12], have analyzed the issue of processing the XPath queries and rewriting the XML queries using views, irrespective of intersection of results of views. Cautis, Deutsch and Onose [13] studied the matter of equivalent rewriting of XPath views without using operators of union or intersection and then they analyzed the effect of applying the intersection operator upon optimization of XPath 2.0 queries. Results of queries are stored in cache as materialized views. Benedikt and others [14] used for XPath queries negation and disjunction operators, which can together simulate an intersection. Another approach [15] consists of replacement of intersection operator with heuristic methods for simplification of XPath expressions.

Let's consider a XML fragment from an XML Web service of transactions SQL Server, for a database 'accounting':

```
<Document name = "accounts">
  < symbol >
     <nr>300</nr>
     <account>RawMaterial </account>
     <DebitTransactions>
     <AssetsLiabilities>A</AssetsLiabilities>
      <value>145000</value>
     </DebitTransactions>
   </ symbol >
   < symbol >
      <nr>401</nr>
      <account> Suppliers </account>
      <DebitTransactions>
      <AssetsLiabilities>L</AssetsLiabilities>
          <value>14000</value>
      </DebitTransactions>
    </ symbol >
</Document>
```

XML data will be represented as a certain tree, unsorted, with a set of edges U, a set of nodes X, a root r and a function f labeling the tree's nodes (Figure 1):

We consider XPath queries containing child nodes marked / and nodes representing descendants' type subtree marked //, predicates used to filter data marked with [ ] and the notation * used for substitution of a descendant node. So, the formalization of the tree representation of an XP (XPath) query uses XP{ /, //, *, [] } language.

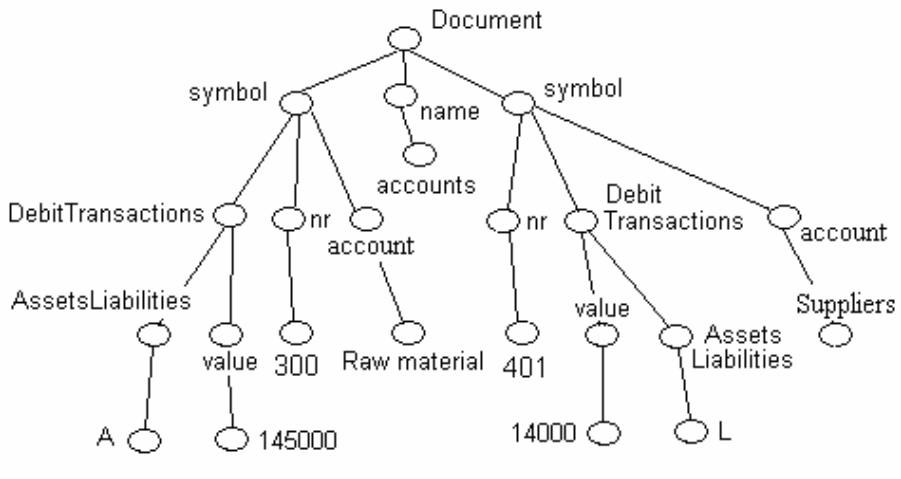

Figure 1. XML tree**.**

Definition of rewriting issue based on XPath views

To define the problem of rewriting of XP views, Cautis, Deutsch and Onose [13] consider one set of V views, defined by XP queries upon a XML document. D_V is the series of document views {doc("v") / v ∈ V}, where the first node is labeled with the name of view. For a query r upon a view document D_V, let mark r' as a query obtained by replacing in r of each view doc("v")/v by the definition of v. Cautis, Deutsch and Onose [13] have described rewriting a view as follows:

Definition 1. Considering a XML document, a XP query marked q and XP views from V series. A plan to rewrite the query q using views from V series is a query r upon D_V with r' ≡ q.

The issue of rewriting a XP view is equivalent to the issue of equivalence of tree models associated to the XP queries classes [16]. If there are defined two models M_1 and M_2, then $M_1 \subseteq M_2 \Leftrightarrow$ there is a plan based on XP views for equivalent rewriting of M_1 using M_2.

3. Tree Models for XPath Queries

Onwards we intend to study some tree models associated to certain XPath queries classes using XP{ /, //, *, [] } representation within rewriting XPath views. We formulate the problem of correspondence of models of trees associated to XPath queries and there are suggested solutions for XPath views rewriting by transformation of tree models.

Author's personal contribution consists in modelling of a set of trees associated to XP queries for a set of constraints of XPath expressions and description of function of correspondence in XP{ /, //, *, [] } language.

Consider XPath query Q_1, as tree representation in Figure 2. Results of Q_1 query for document 'accounts' previously described, are stored in cache as materialized view which extracts information about debit transactions of active asset (A) Raw Material with symbol 300.

Q_1 : // symbol [nr = 300] / DebitTransactions [AssetsLiabilities = "A"]

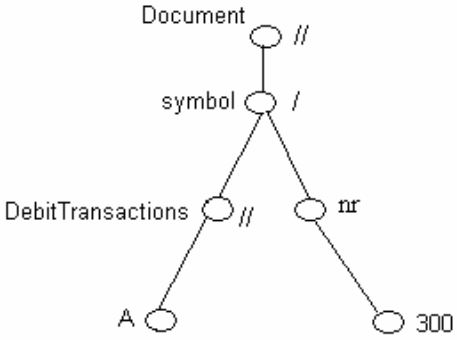

Figure 2. Tree associated to XPath query Q_1

From the above mentioned example, we may generalize and describe by induction a function of correspondence between the set of tree models used for representation of classes of XP queries stored in cache and set of trees associated to XML documents.

Further we introduce the concept of the model of tree associated to XPath query.

Definition 2. A tree model associated to XPath query is defined as a tuple (X_M, U_M, r_M, o_M), where:

- X_M is the set of nodes of model tree,
- U_M is the set of edges,
- r_M is the root,
- o_M contains the XP{ /, //, *, []} representation operators.

Let us note the tree associated to a XML document: (X_G, U_G, r_G), where X_G is the set of nodes of XML tree, U_G is the set of edges, r_G is the root of tree G.

We define the function

$\alpha : X_M \rightarrow X_G$, where X_M is the set of nodes of model tree and X_G is the set of nodes of XML tree, such as each node of model tree has in the XML tree a correspondent:

- for operators /, *, [] , a node and
- for operator //, the root of the subtree extracted from XML tree.

$$\alpha(x) = \begin{cases} y \quad \text{if} \quad x \in X_M \mid O_M - \{//\} \\ \text{the root of XML subtree,} \\ \text{if } x \in X_M \mid // \end{cases} \qquad (1)$$

The result of projection of XP{ /, //, *, [] } representation's operators in the XML document will be found in a subset of nodes from X_G:

$$\alpha(X_M \mid O_M) = \left\{ y \quad / \, y \in X_G^{'} \right\}, \qquad (2)$$

where $X_G^{'} \subseteq X_G$, O_M = {/, //, *, [] }, $X_M \mid O_M$ is the set of nodes of model tree, marked with the operators of the XP{ /, //, *, [] } representation and X_G is the set of nodes of XML tree.

Examples:

$\alpha(X_M \mid *) = \{ x_3 \}$ (Figure 3)

$\alpha(X_M \mid //) = \{ x_4 \}$ Node labeled with operator // corresponds to the node x_4, the root of subtree extracted from XML tree.

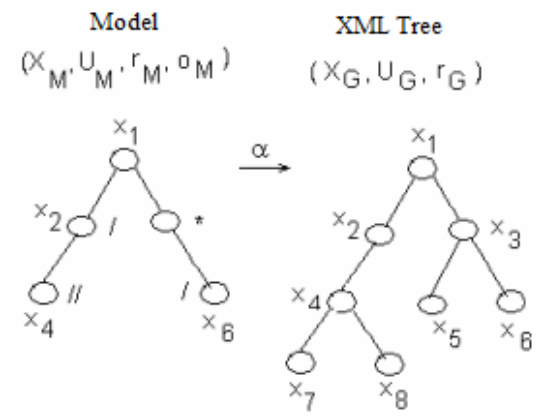

Figure 3. Correspondence between model and XML tree

4. Morphisms of XP models

We will define the correspondence between the tree models used for classes of XPath queries stored in cache and the set of trees associated to a XML document, in the issue of equivalent rewriting of XPath views, in presence of certain constraints $p^{(1)}$, $p^{(2)}$, and so on (predicates used for filtration or validation of XPath expressions).

Rewriting requires filtering of nodes which will form the image equivalent to XP views, from the nodes to be deleted XPath expressions may contain variables for filtration of nodes from trees.

We will take into account that for XP{ /, //, *, [] }, there are many types of representations for the subclasses of XP queries: XP{/, //, [] }, XP{/, //, *} and XP{/, *, [] } (Figure 4):

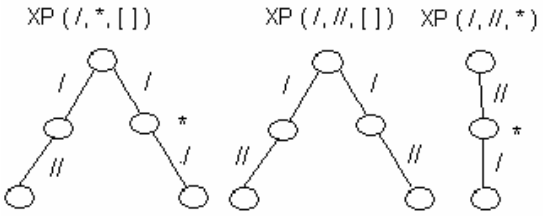

Figure 4. XP models

It has to be solved the issue of correspondence of tree models associated to XPath queries and to check the possibility to rewrite the XPath views by transformation of tree models.

Let be $(X_M^{(i)}, U_M^{(i)}, r_M^{(i)}, o_M^{(i)})$, $i \geq 1$, a model of tree associated to XPath query previously defined. We will consider as follows:

$\alpha^1 : X_M^{(1)} \to X_G$, $\alpha^1(X_M^{(1)} \mid o_M, p^{(1)}) \subseteq X_G$,
where X_G is the set of nodes of XML tree associated to a XML document

We note $X_G^{(1)}$ set of nodes of XML tree associated with XPath view and selected by application of function α^1 to the set $X_M^{(1)}$, with nodes labeled by the operators from o_M and application of $p^{(1)}$ constraint to XPath expressions as follows:

$X_M^{(1)} \mid o_M, p^{(1)} \xrightarrow{\alpha^1} X_G^{(1)}$ (by notation), where $G^{(1)}=(X_G^{(1)}, U_G^{(1)}, r_G^{(1)})$ is a XML subtree, $X_G^{(1)}$ is the set of nodes of XML tree, $U_G^{(1)}$ is the set of edges, $r_G^{(1)}$ is the root

$$\alpha^1(X_M^{(1)} \mid o_M, p^{(1)}) = X_G^{(1)} \quad (3)$$

$X_G^{(1)}$ is the set of nodes of XML tree associated to XPath view because it contains results of the XPath query stored in cache as materialized view.

Similarly, we define the function α^2 and we note $X_G^{(2)}$ the set of nodes of XML trees, associated to XPath view and selected by application of function α^2 to the set $X_M^{(2)}$, with nodes labeled by the operators from o_M and application of $p^{(2)}$ constraint to XPath expressions as follows:

$X_M^{(2)} \mid o_M, p^{(2)} \xrightarrow{\alpha^2} X_G^{(2)}$ (by notation), where $G^{(2)}=(X_G^{(2)}, U_G^{(2)}, r_G^{(2)})$ is a XML subtree, $X_G^{(2)}$ is the set of nodes of XML tree, $U_G^{(2)}$ is the set of edges, $r_G^{(2)}$ is the root.

$\alpha^2(X_M^{(2)} \mid o_M, p^{(2)}) = X_G^{(2)} \subseteq X_G$, and so on.

$X_G^{(i)} (i \geq 1)$ contains results of the XPath query stored in cache, as materialized view.

Further we introduce the concept of XP models.

Definition 3. A model XP is defined as a tuple $G^{(i)}=(X_G^{(i)}, U_G^{(i)}, r_G^{(i)})$, $i \geq 1$, where $X_G^{(i)}=\alpha^i(X_M^{(i)} \mid o_M, p^{(i)})$, $p^{(i)}$ constraints for XPath expressions, $U_G^{(i)}$ is the set of edges, $r_G^{(i)}$ is the root, and $X_G^{(i)}$ contains nodes of XML trees associated to XPath views stored in cache, restricted by application of operators of XP{ /, //, *, [] } representation and by application of $p^{(i)}$ constraints.

Filtering and validation constraints of XPath expressions $p^{(1)}$, $p^{(2)}$, … can be used for selection of nodes of XML trees needed for rewriting XPath views.

We will define the function of transformations in XP{ /, //, *, []} representation:

Definition 4. Considering functions

$\beta_j^i : X_G^{(i)} \to X_G^{(i)}$, $j \geq 1$, where $G^{(i)}=(X_G^{(i)}, U_G^{(i)}, r_G^{(i)})$ is a model XP, $i \geq 1$. We assign β_j^i, $j \geq 1$, functions of transformation of sets of trees associated to XP queries stored in cache, restricted by application of operators of XP{ /, //, *, [] } representation and by application of $p^{(i)}$ constraints, i >= 1, for filtering the XPath expressions.

We note A the set of all functions β_j^i, where

$\beta_j^i : X_G^{(i)} \to X_G^{(i)}$,

$A=\left\{\beta_j^i / \beta_j^i : X_G^{(i)} \to X_G^{(i)}, i, j \geq 1\right\}$

Composition of functions defines a law of composition on A which may be defined as follows: $A \times A \xrightarrow{o} A$,

$(\beta_j^i, \beta_k^i) \xrightarrow{o} \beta_j^i o \beta_k^i, (\forall) \beta_j^i, \beta_k^i \in A, j,k \geq 1$

Verification of the axioms of monoid (A, o):

- composition of functions is associative
- identical function 1_A is the neutral element

Invertible elements of monoid (A, o) are bijective functions $\beta^{(i)} : X_G^{(i)} \to X_G^{(i)}, i \geq 1$, used for equivalent rewriting of XP views (trees with the same number of nodes).

We note $G = \bigcup_{j \geq 1} \beta_j^i, i \geq 1$, the set of all invertible functions of monoid (A, o), used for equivalent rewriting of XP views stored in cache $\Rightarrow$ (G, o) is group

$G = \bigcup_{j \geq 1} \beta_j^i, i \geq 1$, is a set of bijective functions

$\beta_j^i : X_G^{(i)} \to X_G^{(i)}, j \geq 1$

Group (G, o) is commutative: $\beta_j^i, \beta_k^i \in G$, $j,k \geq 1$, β_j^i, β_k^i are functions of equivalent rewriting of a XP query

$\Rightarrow \beta_k^i o \beta_j^i = \beta_j^i o \beta_k^i$

Sentence 1: Consider the set $G = \bigcup_{j \geq 1} \beta_j^i$, where $\beta_j^i : X_G^{(i)} \to X_G^{(i)}$, $j \geq 1$, $G^{(i)} = (X_G^{(i)}, U_G^{(i)}, r_G^{(i)})$ is a model XP, $i \geq 1$. Then on G there is a morphism.

Proof:

We will demonstrate that there is a morphism on the group (G, o):

(G, o) group commutative $\Rightarrow (\forall) f \in G, (\exists) g \in G, f: X_G^{(i)} \to X_G^{(i)}$,

$g: X_G^{(i)} \to X_G^{(i)}$, $i \geq 1$, such as

$f \circ g = g \circ f = 1_G$

We define the function $\varphi: G \to G$, $\varphi(h) = fohog$, $h \in G$

We have to demonstrate that φ is a morphism of groups:

$\varphi(h \circ h') = \varphi(h) o \varphi(h')$, $(\forall)$ h, $h' \in G$

$\varphi(h \circ h') = f \circ h \circ h' \circ g = (f \circ h \circ g) \circ h' = (f \circ h \circ g) \circ h' \circ 1_G =$

$= (f \circ h \circ g) \circ (h' \circ f \circ g)$ (4)

$\varphi(h) \circ \varphi(h') = (f \circ h \circ g) \circ (f \circ h' \circ g) = (f \circ h \circ g) \circ (h' \circ f \circ g)$ (5)

From the relations (4) and (5) $\rightarrow \varphi(h \circ h') = \varphi(h) \circ \varphi(h') \Rightarrow \varphi$ is a morphism on $G = \bigcup_{j \geq 1} \beta_j^i$, $i \geq 1$.

Comments: The existence of a morphism of XP models proves the possibility of equivalent rewriting of XPath views, using transformations of tree models associated to XP queries from cache, under filtering constraints of XPath expressions, in XP{ /, //, *, [] } representation.

5. Semantic Cache of XP Views

We consider a view V materialized in cache and a XPath query Q which must be processed. Composition of queries using a semantic cache of XPath views, supposes the existence of a query C which, by composition with a view V from cache, will return the result of query Q, that is C o V = Q.

Within context of composition of queries using the cache of XP views, we will describe the operation of inserting a XP view in cache, the selection of a view from cache and then we will present a case when a query cannot be resolved from the cache, even if its results are already there.

Insertion of View in Cache

We consider XP views V_i, $1 \leq i \leq n$ and the filtering and validation constraints of XPath expressions $p^{(1)}$, $p^{(2)}$, and so on.

A semantic description of cache with n XP views, may be as follows:

semantic cache **:=** { V_i / V_i **:** / a / child **::** b [$p^{(j)}$] [$p^{(k)}$], where a and b are XML elements of view, $1 \leq i \leq n$, j, $k \geq 1$}

Inclusion of XPath expressions $p^{(i)} \leq p^{(j)}$, i, j ≥ 1, shows that the set of nodes resulted from the evaluation of $p^{(i)}$ is included in the set of nodes selected by the $p^{(j)}$ expression. A XPath expression is considered invalid when the set of nodes evaluated is always void.

Mandhani and Suciu [7] have stored cache of XPath views in tables of a Microsoft SQL Server relational database, which offers XML support. Cache is stored in three relational tables, one containing a column type XML, and the other 2 tables memorize the prefix common to XPath views and queries and the conditions of XML elements filtering.

Lee and Wesley Chu [8] have stored the semantic cache in Hash tables with input data of type (key, value), where the key is a semantic description based on the precedent queries and the value contains results of queries associated to the key.

A semantic cache of XPath views can be created using a collection of TigerLogic XDMS files, from Stylus Studio XML Enterprise Suite, stored in the Microsoft SQL Server relational database. XQuery queries will use as source the cache represented by the TigerLogic XDMS files collection and will be executed by the TigerLogic XDMS XQuery processor.

Selection of XP view from cache

For use of XP view from cache, let's consider three XP views and a query Q:

V_1**:** / a / child **::** b [x = "sir"] [y >= 13000]
V_2**:** / a / b [x = " sir "]

V_3**:** // b [x = " string"] /c

Q**:** // child **::** b [x = "sir"] [y >= 10000] / c

Query Q is related to views V_1 and V_2 (results of query Q are included in views V_1 and V_2), but the constraint [y >= 13000] does not contain [y >= 10000]. Therefore only view V_2 returns results for query Q, and query C, to whom it composes is:

C**:** // b [y >= 10000] / c

Composition of queries complies with relation C o V_2 = Q.

This is a demonstration that to process the query Q, the cache of XPath views may be used, because there is a query C and a view V_2 materialized in cache thus C o V_2 = Q.

A special case is when a query cannot be resolved from the cache, even the semantic cache contains materialized views.

Let's consider a query Q and two XP views V_1 and V_2 stored in cache.

Q**:** / a / b [x = "sir"] [y >= 10000] / c

V_1**:** / a / b [x = "A"]

V_2**:** // child **::** b [y <= 3000]

Results of query Q are included in views V_1 and V_2, but the constraints [y >= 10000] and [x = "sir"] (for node b) are not related to views V_1 and V_2.

6. Conclusions

This paper presents a study of issue of equivalent rewriting of XPath queries, using XP views stored in cache.

The author's personal contribution consists of modelling the set of trees associated to XP queries and description of the functions of transformation in XP{ /, //, *, [] } language, for a set of constraints of XPath expressions.

An original approach is Sentence 1, showing that there is a morphism on the group of XP models, suggesting the possibility to rewrite the XPath queries, by transformation of tree models associated to XP views, under filtering constraints of XPath expressions, in XP{ /, //, *, [] } representation.

It demonstrates that a semantic cache of XPath views may be used for processing a XPath query Q, because there is a query C and a view V materialized in cache thus CoV=Q.

REFERENCES

1. KOSSMAN, D., D. FLORESCU, **Storing and Querying XML Data using an RDBMS**, IEEE Data Engineering Bulletin, 1999.
2. ZHANG, C., J. NAUGHTON, D. DEWITT, Q. LUO, G. LOHMAN, **On Supporting Containment Queries in Relational Database Management Systems**, in ACM SIGMOD, 2001, pp. 425-436.
3. YOSHIKAWA, M., A. TOSHIYUKI, T. SHIMURA, S. S. UEMURA, **Xrel: A Path-based Approach to Storage and Retrieval of XML Documents using Relational Databases**, ACM Transactions on Internet Technology, Nr. 1, 2001, pp. 110-141.
4. KRISHNAMURTHY, R., R. KAUSHIK, J. F. NAUGHTON, **XML to SQL Query Translation Literature: The State of the Art and Open Problem**, in XML Database Symposium, XSym, 2003, pp. 31-38.
5. BOHANNON, P., J. FREIRE, P. ROY, J. SIMEON, **From XML Schema to Relations: A Cost-based Approach to XML Storage**, in 18th International Conference on Data Engineering, 2002, pp. 64-76.
6. LUO, Q., S. KRISHNAMURTHY, C. MOHAN, H. PIRAHESH, H. WOO, B. LINDSAY, J. NAUGHTON, **Middle-tier Database Caching for e-Business**, in Proceedings of the ACM SIGMOD, 2002, pp. 600 – 611.
7. MANDHANI, B., D. SUCIU, **Query Caching and View Selection for XML Databases**, Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005.

8. DONGWON, L., W. CHU WESLEY, **A Semantic Caching Scheme for Wrappers in Web Databases**, In ACM International Workshop on Web Information and Data Management (WIDM), USA, 1999.
9. TANG, J., S. ZHOU, **A Theoretic Framework for Answering XPath Queries using Views**, In XSym, 2005.
10. BALMIN, A., F. ÖZCAN, K. S. BEYER, R. COCHRANE, H. PIRAHESH, **A Framework for using Materialized XPath Views in XML Query Processing**, in VLDB, 2004.
11. LAKSHMANAN, L. V. S., H. WANG, Z. ZHAO, **Answering Tree Pattern Queries using Views**, in VLDB, 2006.
12. GAO, J., T. WANG, D. YANG, **MQTree based Query Rewriting over Multiple XML Views**, in DEXA, 2007.
13. CAUTIS, B., A. DEUTSCH, N. ONOSE, **XPath Views for Documents with Persistent Identifiers**, in SIGMOD, 2007.
14. BENEDIKT, M., W. FAN, F. GEERTS, **XPath Satisfiability in the Presence of DTDs**, in PODS, 2005.
15. GROPPE, S., S. BÖTTCHER, J. GROPPE, **XPath Query Simplification with Regard to the Elimination of Intersect and Except Operators**, in ICDE Workshops, 2006.
16. XU, W., Z. M. ÖZSOYOĞLU, **Rewriting XPath Queries Using Materialized Views**, VLDB 2005.
17. VOLOVICI, D., G. D. CUREA, M. BREAZU, D. I. MORARIU, **Statistical Methods for Performance Evaluation of WEB Document Classification**, Studies in Informatics and Control, Vol. 19, No. 2, 2010.