International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 5– No. 5, April 2013 – www.ijais.org

Frequent XML Query Caching in ebXML based E-commerce Applications using Association Rule Mining

Anju Vijayan
PG student
Pondicherry University
India

V. Uma
Assistant professor
Pondicherry University
India

G. Aghila,Ph.D.
Professor
Pondicherry University
India

ABSTRACT
Mining frequent XML query patterns and caching the results can improve the performance of XML based systems. Temporal features of queries can be used to guide cache replacement. Many approaches have been proposed to mine frequent XML query patterns and caching the results. Those approaches mine frequent subquery patterns from historical queries. But for ebXML (electronic business using XML) based applications in e-commerce, most of the queries will be of same structure. In such cases instead of mining subqueries, frequent queries can be mined directly. In this work an efficient system for frequent XML query caching in ebXML based e-commerce applications is proposed which incorporates temporal features of frequent queries for cache replacement. Association rules are mined between frequent queries to discover temporal patterns among them. Semantically closer infrequent queries are clustered and their associations with frequent queries are also mined. These rules are used to guide cache replacement so that subsequent query results will not get replaced from cache.

General Terms
Frequent XML query caching, ebXML, Association rule mining.

Keywords
Frequent XML query mining, Frequent XML query caching, ebXML.

1. INTRODUCTION
XML is widely used for data representation and exchange over internet. Many e-commerce applications use XML for data exchange because of its self describing property and semi-structure nature. The drastic use of XML data in e-commerce applications amplified the significance of frequent XML query caching. ebXML is a modular suite of specifications for business parties to exchange data in e-commerce. Companies using those specifications have standard data frame to exchange messages and data in e-commerce[1][8][9].

The existing methods[4][2][3][5] for caching frequent XML queries may not be suitable for ebXML based applications since the frequent pattern mining algorithms used by those approaches are not utilizing the structural similarity[1][9] of ebXML queries and degrades performance. Those approaches [4][2][3][5] mine frequent subtrees using conventional generate and test strategy. Candidate subtrees are generated from all query trees and their support is discovered by comparing them with all query trees. This approach is not required for ebXML based applications in e-commerce where majority of XML queries have identical structure [1]. Therefore ebXMiner developed by Chang et al. [1] is used by this work to discover frequent queries in single database scan. ebXMiner first collects identical queries and then finds their supports. Frequent subqueries are generated only from infrequent queries.

For cache replacement, this work considers temporal features of frequent queries to avoid eviction of subsequent query results from cache. Association rules are mined among frequent queries to predict which query will probably come next. Chen et al. [2] proposed a method for mining association rules to design a rule based LRU for cache replacement. However, [2] is not suitable for ebXML based applications in e-commerce since they are not exploiting the fact that most of ebXML queries will be identical [1]. Approach used in [2] clustered semantically closer queries together and association rules are mined among those clusters because direct association among queries may not be frequent always. In ebXML based applications most of the queries are identical and hence most of the queries are frequent. Therefore this work mines association rules directly between frequent queries. Infrequent queries are not avoided completely. Infrequent queries are clustered based on containment of frequent sub query part. Then association rules are discovered from those clusters and frequent queries because after / before a frequent query, similar type of infrequent queries may come. Since this work considers direct associations among frequent queries, subsequent query prediction will be more precise. A modified LRU called frequent query association based LRU (FA-LRU) is introduced which utilizes the mined associations for cache replacement so that subsequent query results will not get replaced.

Rest of the paper is organized as follows: Section 2 discusses related works about frequent XML query mining and caching. Section 3 describes the new approach for XML query caching in ebXML based e-commerce applications. Section 4 illustrates comparative study of the proposed approach with existing approaches. Section 5 includes conclusion and future work. Section 6 contains references of this work.

2. RELATED WORKS
There are some works to mine frequent xml queries and caching the results [4][2][3][5][6]. Yang et al. [4] introduced an algorithm called FastXMiner to mine frequent XML queries and extended LRU based on those frequent query patterns. Li et al. [3] proposed an algorithm which finds frequent XML queries from evolving databases. Hua et al. [5]

developed QPTMiner algorithm to discover frequent XML query patterns for cache replacement. Bei et al. [6] introduced a bottom up approach to mine frequent rooted subtrees for XML query caching and web access pattern mining. However, these works [4][3][5] are not incorporating temporal patterns of frequent queries for cache replacement and frequent query mining algorithms used by those approaches are not utilizing the structural similarity of ebXML queries. This work uses ebXMiner [1] which mines frequent queries by considering structural similarity of ebXML queries.

In this work, association rules are mined to discover temporal features of queries. Chen et al. [2] proposed a method for mining positive and negative association rules for designing a rule based LRU for XML query caching. However, [2] may not be suitable for ebXML based e-commerce applications since they discover frequent subqueries using traditional generate and test strategy which is time consuming. Then [2] clustered queries based on the semantics. Association rules are mined among the clusters. Here in this work association rules are discovered directly between frequent queries. Clustering is done only for infrequent queries based on containment of frequent subqueries. Then association rules are mined among those clusters and frequent queries. These rules are used for cache replacement. Since this work makes use of direct association among frequent queries, subsequent query prediction will be more accurate.

3. PROPOSED SYSTEM

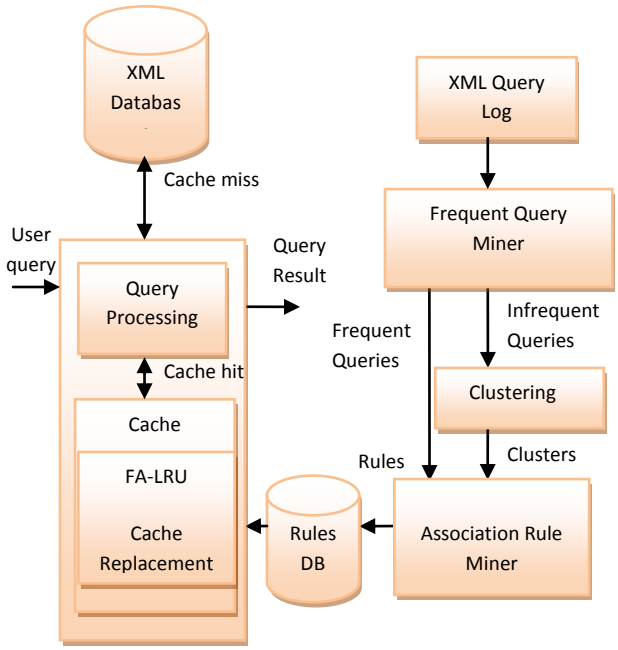

Figure 1: System Architecture

In this work frequent xml queries are mined from historical queries using ebXMiner. Frequent subqueries are mined from infrequent queries. Then infrequent queries are clustered based on the containment of frequent subqueries. After that association rules are mined among the frequent queries and infrequent query clusters. These rules are used for predicting queries which may come next while cache replacement.

XML queries are represented as trees as shown in Figure 2 since the XML data has the tree structure.

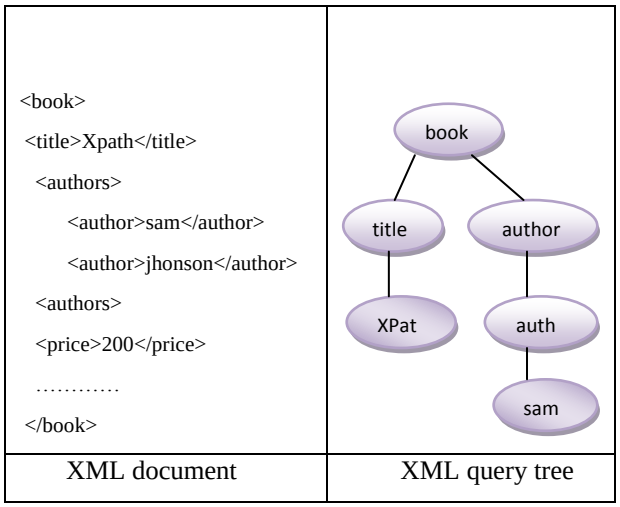

Figure 2: XML Query Tree

Consider a query database D of 12 queries Q1, Q2, Q3... Q12. Table 1 shows the sequence of queries at time instant T1 and T2. ebXMiner mines frequent query trees and frequent rooted subtrees of infrequent queries. Infrequent queries are grouped into clusters labeled by frequent rooted subtrees to collect semantically closer infrequent queries together. Suppose Q1, Q2 and Q3 in figure 3 are infrequent queries and t1 is a frequent rooted subtree. Then Q1 and Q2 are added into a cluster labeled by t1 since they both contain t1. If any infrequent query does not contain any frequent rooted subtree, that query is discarded. If an infrequent query contains more than one frequent rooted subtrees, it is added to the cluster labeled by larger rooted subtree. For example Q3 contains two frequent rooted subtrees t2 and t3. Therefore Q3 is added to cluster labeled by t3.

If two subtrees are of same length, support is taken into consideration. Query is added to the cluster corresponding to the subquery with larger support. If support is also same, cluster dissimilarity metric used in [2] is employed to decide cluster to which the query should be added. In their work cluster dissimilarity is measured in terms of number of infrequent edges in the cluster. A merged tree is generated from all the trees in the cluster and inter cluster dissimilarity is measured as the ratio of number of infrequent edges in the merged tree to total number of edges in the merged tree. Queries are added to cluster with minimum inter cluster dissimilarity.

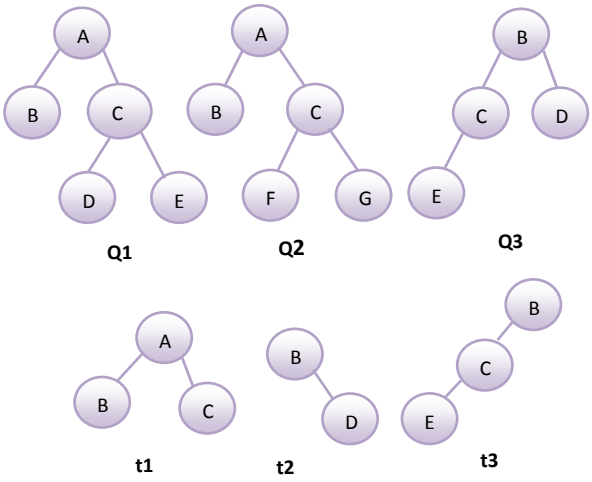

Figure 3

Once infrequent queries are clustered next step is to replace them with their corresponding clusters. Suppose Q1, Q2, Q7, Q4, Q9 are infrequent queries and Q1, Q2 belong to cluster C1. And Q7, Q9 belong to cluster C2. Replace the infrequent queries with their corresponding clusters as shown in table 2. Q4 does not belong to any cluster so it is discarded from rule mining.

Table 1

ID	T1	T2
1	Q3	Q5
2	Q6	Q8
3	Q8	Q2
4	Q10	Q12
5	Q7	Q10
6	Q12	Q4
7	Q1	Q11
8	Q11	Q9
9	Q5	Q6
10	Q3	Q8
11	Q10	Q6
12	Q5	Q12

Table 2

ID	T1	T2
1	Q3	Q5
2	Q6	Q8
3	Q8	**C1**
4	Q10	Q12
5	**C2**	Q10
6	Q12	#
7	**C1**	Q11
8	Q11	**C2**
9	Q5	Q6
10	Q3	Q8
11	Q10	Q6
12	Q5	Q12

3.1 Rule Mining

Next step is to mine positive and negative association rules between frequent queries and infrequent query clusters. In this work support-confidence-interest framework [7] [2] is used for association rule mining. For database D as shown in table 2, Positive association rule is of the form X→Y where

Support (X → Y) ≥ min_support &
Confidence (X →Y) ≥ min_confidence &
Interest (X → Y) ≥ min_interest.

Support (X → Y) is fraction of sequences in database D in which X is followed by Y.
Confidence (X→Y) = Support (X → Y) / Support (X, _).
Interest (X → Y) =Support (X → Y) / Support (X, _) Support (_ ,Y).
This work considers negative association rules of the form X → ¬Y where

Support (X → ¬Y) ≥ min_support &
Confidence (X → ¬Y) ≥ min_confidence &
Interest (X → ¬Y) ≥ min_interest.

Support (X →¬ Y) is fraction of sequences in database D in which X is not followed by Y.
Confidence (X→¬Y) = Support (X → ¬Y) / Support (X, _).
Interest (X → ¬Y) =Support (X → ¬Y) / Support (X, _) Support (_, Y)

This work makes use of
frequent_query→frequent_query
frequent_query→infrequent_query_cluster
infrequent_query_cluster→frequent_query

infrequent_query_cluster→infrequent_query_cluster
frequent_query→¬frequent_query
frequent_query→¬infrequent_query_cluster
infrequent_query_cluster→¬frequent_query
infrequent_query_cluster→¬infrequent_query_cluster
associations for cache replacement.

Algorithm used for rule generation is similar to the work done by Chen et al. [2]. But here in this work frequent 1 – item generation is avoided since database contains only frequent queries. Only for infrequent query clusters, support checking is done. In [2] association rules are mined among clusters of semantically closer queries. This work considers associations between frequent queries and infrequent query clusters.

3.1.1 Algorithm for rule generation

Input: Data base D of frequent queries and infrequent query clusters, min_support, min_confidence, min_interest

Output: RPos : Set of positive association rule, RNeg : Set of negative association rule

Method:

(1) Scan database D and discard clusters C
with Support(C) < min_support

(2) E = 1-item sequence of Database D

(3) K= E × E /* candidate frequent 2- item sequence of database D */

(4) for each item (k_i, k_j) ∈ K

(5) if(Support ($k_i \rightarrow k_j$) ≥ min_support then

(6) if(Confidence ($k_i \rightarrow k_j$) ≥ min_confidence &&

(7) Interest ($k_i \rightarrow k_j$) ≥ min_interest

(8) $Rpos = Rpos \cup \{k_i \rightarrow k_j\}$

(9) end if

(10) end if

(11) else

(12) if(Support ($k_i \rightarrow \neg k_j$) ≥ min_support then

(13) if(Confidence ($k_i \rightarrow \neg k_j$) ≥ min_confidence &&

(14) Interest ($k_i \rightarrow \neg k_j$) ≥ min_interest

(15) $RNeg = RNeg \cup \{ k_i \rightarrow \neg k_j \}$

(17) end if

(18) end if

(19) end for

3.2 Cache Replacement

In this work a modified LRU is introduced by incorporating temporal features of historical queries. Association rules are mined from frequent queries and semantically closer infrequent queries. Association rules are used for predicting queries which may come next. Each time when a new query comes most recent value R_{top} is incremented by one and rule set is checked for the rules containing Q_i on left hand side of the rule. If no such rules, checking is done to find whether Q_i is contained by any cluster. If Q_i is present in any cluster, checking is done to find the rules corresponding to that cluster. If any rule of the above mentioned types is found, replacement value of the corresponding queries/cluster is updated.

If any rule of type $Q_i \rightarrow Q_j$, $Q_i \rightarrow C_k$, $C_i \rightarrow C_k$, $C_i \rightarrow Q_j$ is found, new replacement value of right hand side query or cluster is updated as R'= R + (R_{top} - R) * confidence of the rule. Since $R \leq R' \leq R_{top}$, eviction of queries which may come next is delayed. Similarly if any rule of type $Q_i \rightarrow \neg Q_j$, $Q_i \rightarrow \neg C_k$, $C_i \rightarrow \neg C_k$, $C_i \rightarrow \neg Q_j$ is found new replacement value of right hand side query or cluster is updated as R' = R - (R_{top} - R) * confidence of the rule. Since R'< R, priority of such queries are reduced and it speeds up the removal of such queries from the cache. This strategy is similar to the work of Chen et al [2]. But [2] followed a generalized approach which considers associations between clusters of semantically closer queries for cache replacement. In this work more powerful set of rules are used for cache replacement which considers inter frequent query, infrequent query cluster- frequent query, frequent query-infrequent query cluster and inter infrequent query cluster associations. Hence subsequent query prediction is expected to be more accurate and cache replacement will be more efficient. As a result, average response time can be reduced by significant amount of time.

4. COMPARITIVE STUDY

This section compares the new approach FA_LRU with existing techniques LRU_FQPT [2], LRU_AR [4] for frequent XML query caching.

Table 3: Comparison between LRU_FQPT, LRU_AR and FA_LRU.

Serial No	Method	Description	Advantage	Disadvantage
1	LRU_FQPT	Frequent subqueries are mined and their results are cached to reduce query response time.	More efficient compared to traditional LRU cache replacement since cache miss is less compared to LRU.	Temporal features of frequent queries are not considered
2	LRU_AR	Frequent subqueries are mined and clustered queries based on frequent subquery part. Association rules are	Temporal patterns among queries are considered for cache replacement to avoid eviction of	Not appropriate for ebXML based applications since structural similarity of ebXML

		mined between the query clusters to assist query replacement	frequent subsequent query result. Query response time and cache miss is less compared to LRU_FQPT and LRU.	queries is not considered for frequent query mining.
3	FA_LRU	Frequent queries are mined directly and infrequent queries are clustered based on frequent subquery part. Association rules are mined among frequent queries and infrequent query clusters to assist cache replacement.	More efficient in ebXML based applications compared to LRU_AR, LRU_FQP and LRU since query response time expected to be less compared to them.	Not a generalized approach.

Table 4 contains a brief comparison of FA_LRU with existing methods. For fixed cache size and number of queries ranges from 100-500, average response time (in seconds) is expected to be varied as shown in Figure 4. Average response time is calculated as the ratio of total execution time for answering a group of queries to total number of queries in the group.

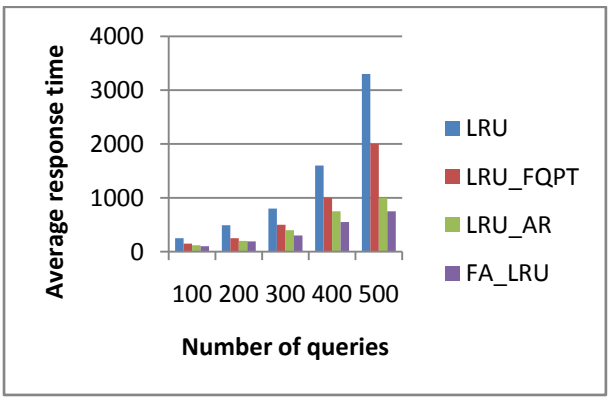

Figure 4: No: of queries Vs Average response time graph.

5. CONCLUSION AND FUTURE WORK

In this work an efficient frequent XML query caching approach is introduced for ebXML based e-commerce applications. Since this work considers temporal features of frequent queries, prediction will be more precise and average response time is expected to be reduced by significant amount of time. This work also utilizes frequent subquery patterns of infrequent queries. Infrequent queries are clustered and their association with frequent queries is also considered.

Currently this work mines binary associations among queries. In future sequential relations among queries will also be considered.

6. REFERENCES

[1] T. Chang, S. Chen, Frequent Xml Query pattern mining for ebXML applications in Ecommerce, Expert Systems with Applications, An International journal archive, Volume 39 Issue 2, February 2012, Pages 2183-2193.

[2] L. Chen, S. S. Bhowmick, L. T. Chia, Mining Positive and Negative Association Rules from XML Query Patterns for Caching, Proceedings of the 10th international conference on Database Systems, 2005, Pages 736-747.

[3] G. Li, J. Feng, J. Wang, Y. Zang, L. Zhou, Incremental pattern mining from xml queries for caching, Data Mining, Sixth International Conference on Computing & Processing Hardware /Software, 2006, Pages 350-361.

[4] L. H. Yang, M. L. Li, W. Hsu, Efficient mining of XML query patterns for caching, Proceedings of the 29th international conference on Very large data bases, 2003, Pages 69-80.

[5] C. Hua, Frequent Query Patterns Guided XML Caching and Materialization, Wireless Communications Networking and Mobile Computing, International Conference on Communication Networking Broadcasting, 2007, Pages 3673-3676.

[6] Y. Bei, G. Chen, L. Shou, X. Li, J. Dong , Bottom up discovery of frequent rooted unordered subtrees, Information sciences, Volume 179, Issues 1-2, 2 January 2009, Pages 70-88.

[7] X. Wu, C. Zhang, S. Zhang, Mining both positive and negative association rules. In Proc. of ICML, 2002, Pages 381-405.

[8] Jin, Z. Yong, Ye, S. Ping, ebXML compatible agent communication language, Proceedings of international conference on computational and information sciences, 2011, Pages 361-365.

[9] ebXML. Available from: http://www.ebxml.org/