Prefetching Web Objects with Inter-Query Relationships

SHIU HIN WANG, VINCENT NG
Department of Computing
The Hong Kong Polytechnic University
Hung Hom, Kowloon
HKSAR

Abstract: - By virtue of the prevalence of XML in data interchange and databases publishing, XML queries become more popular employed in enquiry of XML documents. Nowadays many researches have been done to discover the queries patterns. Very often the patterns are value specifics and neglect the underlying relationship. In this paper, we present a framework to support prefetching of web objects through mining of query relationships. We start with a presentation of a method to identify the possible relationships between XML queries. With a query log, sessions of enumerated query relationships can be formed. After mining of the sessions, query patterns are discovered and can be utilized for prefetching so as to improve query performance.

Key-Words: - XML, Prefetching, XML Query

1 Introduction

As the web logs and transaction files from web databases provide queries histories issued from clients, analyzing of the transaction histories will provide insight users patterns for various purposes. In some researches, the discovery of these relationships enables the caching and prefetching for OLTP of DBMS and web objects. For example, mining of queries from transaction logs and URLs in web server log provides a means of data sources.

The presence of structures in XML documents poses a new challenge for retrieval of data. Various works[18,20,21] have been done on flexible/approximate query answering through query relaxation and definitions of new indexing or ranking models against XML documents. In addition, the study of query matching[12], containment problem[9] and exploitation of XPath[1] in storage of fragmented query results[10] raise the interests to develop efficient replacement strategies. Apart from passive caching of query results, prefetching is also exploited by [3,5,8,13,14,15] to improve the user perceived latency.

Inter-query relationships represent the patterns existing in consequent data queries. In traditional object and relational database systems, we may identify the relationship through the investigation of result sets or the queries themselves.
Regarding to XML, Inter-query relationships may refer to the data queries applied to different XML documents. To simplify our investigation, we make the following assumptions:

1) Only one XML document instance
2) Queries are 'trivial' such that the result are either a document node or a subtree rooted at one node of the original document.

The first assumption assumes that only one XML document is considered. We assert the assumption because the schema and contents between two different DTDs may be different and therefore the inter-relationships of queries will be meaningless. In other words, the structure and semantics of the data will not be utilized. For the second assumption, we only handle those 'trivial' queries such that the results are either a document node or nodes subtree. In other words, predicates other than the leaf must evaluate to at least one node.

In this paper, we will illustrate the identification of inter-query relationship. Although many researches have been done on the study of inter-query relationships of SQL transactions and URLs of Internet queries, URLs and virtual path of web objects only represent the nature, the location of requested objects and design of the system. By contrast, XML is different in nature from RDBMS and Web in its inherent hierarchical and semantics information.

In our work, we are interested to extend the StructCache framework[23] to support prefetching of XML queries through mining of inter-query relationships. We start with a presentation of a

method to identify the possible relationships between XML queries. With a query log, sessions of enumerated query relationships can be formed. After mining of the sessions, query patterns are discovered and can be utilized for prefetching so as to improve query performance.

Section 2 describes the different representations of XML queries and mentions Inter-query relationships under specific assumptions and measures. Mining of inter-query relationships is then covered in Section 3.
Section 4 presents how the mining can be applied to the StructCache framework[23]. In addition, it also describes how the modifications are done in the internal architecture of the framework to support the prefetching of web objects. Lastly, Section 5 concludes our work.

2 Inter-Query Relationships

It is generally true that queries are not purely random. In many cases, users have particular patterns when asking questions. In this section, we are interested in patterns in the querying answering.
Inter-Query relationship refers to the patterns existing in consequent data queries. For a pair of queries expressed in XPath expression, Q1 and Q2, we like to represent the possible inter-relationship of them. In this paper, five possible relationships are enumerated, namely Γ_1, Γ_2, Γ_3, Γ_4, Γ_5. Before proceeding to define the different relationships, we will illustrate a XPath query representation method using one suggested encoding scheme in the coming section. Section 2.2 will be devoted to define basic measures that can help us elaborate different relationships among nodes in a tree and section 2.3 covers the definitions of the inter-query relationships.

2.1 Query Representation

It is well known that a XML document can be expressed as a tree. An example XML instance is shown in Figure 1 and the following XPath queries are asked against it.

Q1: /PLAY
Q2: /PLAY/PERSONA
Q3: /PLAY/ACT
Q4: /PLAY[TITLE='Hamlet']/ACT/TITLE
Q5: /PLAY[TITLE='Hamlet']/ACT[.='Act II']/SCENE

The XPath expressions have not considered the structural relationship between the nodes in the XML document. Suppose the relative position between two XPath indicate the distance between them and absolute position represents their locations with respect to the root node. A simple method to capture both the absolute and relative positions of the nodes in the XML queries is by using subscripts to augment the node labels.

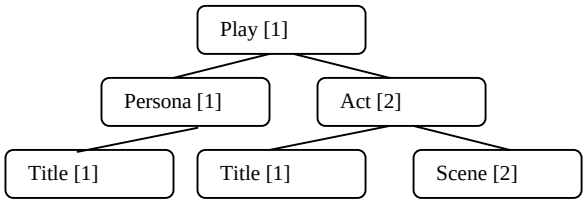

Fig. 1 Tree representation of an XML document

In Fig. 1, each node is assigned an index ranged from 1 to n, where n represents the number of children of their corresponding parents. A node is then identified by traversing the path from the root node and concatenating the augmented index until reaching itself. In other words, the 5 queries are represented as:

$Q_1 \rightarrow Q_{(1)}$
$Q_2 \rightarrow Q_{(1,1)}$
$Q_3 \rightarrow Q_{(1,2)}$
$Q_4 \rightarrow Q_{(1,2,1)}$ {constraints}
$Q_5 \rightarrow Q_{(1,2,2)}$ {constraints}

The assignment of subscripts requires the structural knowledge from the XML file. The enumeration has the following four characteristics:

1) Each node has an unique identity to indicate it's absolute position within the XML tree;
2) The horizontal position among the siblings can be inferred by the last index(dimension) of the subscript;
3) The depth/level(vertical) of nodes are indicated by "dimensions" of the subscript. In other words, number of values represent the longest possible location path; and
4) Relative position of two selected nodes can be determined.

2.2 Primitive Measures

This section defines five measures which will be used for our discussion of inter-query relationships in section 2.3.

Definition 1
Length(Q) : for a given query $Q_{(i1,i2,\ldots ik)}$ {constraints}, Length (Q) is k.

Definition 2
VerticalDifference(Q_1 ,Q_2) or $D_V(Q_1, Q_2)$: it is defined as **|Length(Q_1) - Length(Q_2)|**.

Definition 3
Suffix(Q_1 ,Q_2) : it is the maximum length(number of nodes) of sub-string overlapping between Q_1 and Q_2 counting from the last node in a reverse order. In addition, 0 <= **Suffix(Q_1, Q_2) <= max(Length(Q_1), Length(Q_2))**.

Definition 4
Prefix(Q_1 ,Q_2) : it is the maximum length(number of nodes) of sub-string overlapping between Q_1 and Q_2 counting from the beginning. In addition, 0 <= **Prefix(Q_1, Q_2) <= max(Length(Q_1), Length(Q_2))**.

Definition 5
Dist(Q_1, Q_2) = Length(Q_1) + Length(Q_2) - 2 * Prefix(Q_1, Q_2)

Definition 5 denotes the difference nodes from the query results in a XML tree. In short, the distance is the number of edges traversed from one node to another. That is, it is the 'path distance' between the two queries.

Here, we use two examples to illustrate the first four measures for our sample queries in section 2.1:

Case 1: Q_2 => Q_4
/PLAY/PERSONA => /PLAY/ACT/TITLE
D_V (Q_2,Q_4) = 1, Suffix(Q_2,Q_4) = 0, Prefix(Q_2,Q_4) = 1.

Case 2: Q_4 => Q_5
/PLAY[TITLE='Hamlet']/ACT/TITLE => /PLAY/ACT/SCENE
D_V (Q_4,Q_5) = 0, Suffix(Q_4,Q_5) = 0, Prefix(Q_4,Q_5) = 2.

The Path Equivalence Class (PEC) takes advantages of the semantic context by separating the tree nodes from data leaves[21]. The maximal prefix defined in [21] is similar to our function **Prefix()** defined. However, our encoding scheme does nothing with the supporting of XML document indexing nor the navigating algorithms for retrieval of approximate query embeddings. Instead, it exploits the index path (XPath) and the reformed expression to assist the definition of inter-query relationships. In addition, it also provides hints about user patterns and query heuristics.

2.3 Types of Inter-Query Relationships

Here, we classify the inter-query relationships into the five different types in accordance with the measures already defined:

(1) Self(G_1)
The Self() relationship, shown in Fig. 2, represents the two queries, Q_1 and Q_2, access the same node. It implies the following criteria are satisfied:

1. $D_V(Q_1, Q_2) = 0$
2. Prefix(Q_1, Q_2) = Suffix(Q_1, Q_2)

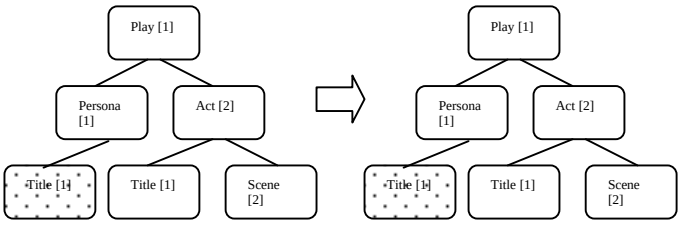

Fig. 2 Relationship G_1

(2) SamePath(G_2)
The SamePath(), as shown in Fig. 3, is a type of relationship that one encoded path is a sub-path of the other and both are rooted at the same node. In other words, the following criteria must be satisfied:

1. $D_V (Q_1, Q_2) \neq 0$
2. Prefix(Q_1, Q_2) = min{Length(Q_2), Length(Q_1)}
3. Suffix(Q_1, Q_2) = 0

Hence, the parent-child and grandfather-grandchild relationships are specific cases of Γ_2. For direct parent relationship, $D_V (Q_1, Q_2)$ has a value equal to 1. For the 'Grandfather-Grandchild' relationship, $D_V (Q_1, Q_2)$ has a value greater than 1.

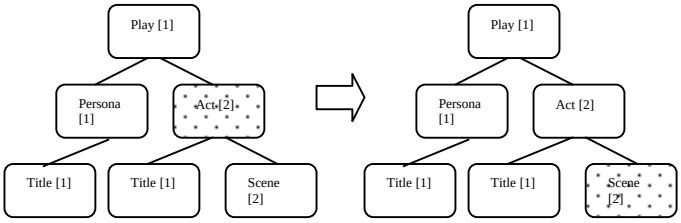

Fig. 3 Relationship G_2

(3) Sibling(G_3)
The Sibling(), as shown in Fig. 4, is a type of relationship the node of the preceding query is a sibling that of the following query. The following criteria will be satisfied for this kind of relationship:

1. $D_V(Q_1, Q_2) = 0$
2. $Prefix(Q_1, Q_2) = Length(Q_2) - 1 = Length(Q_1) - 1$
3. $Suffix(Q_1, Q_2) = 0$

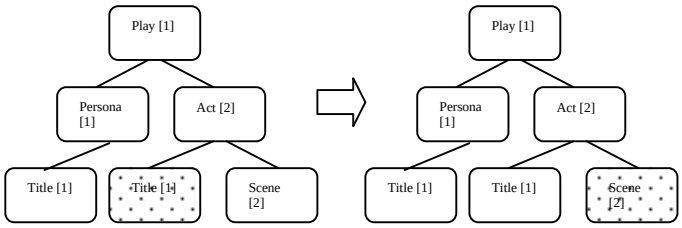

Fig. 4 Relationship G_3

(4) Ancestor-Only(G_4)

The Ancestor-Only(), as shown in Fig. 5, represents the two nodes of the queries have a common ancestor relationship but not the parent nor the same path of each other. In other words, the queries are of different levels in the hierarchy and the following criteria must be satisfied:

1. $D_V(Q_1, Q_2) \neq 0$
2. $Prefix(Q_1, Q_2) > 0$
3. $Prefix(Q_1, Q_2) < min\{Length(Q_2), Length(Q_1)\}$

Therefore, 'Uncle-Nephew' relationship is a specific case of Γ_4 whenever $D_V(Q_1, Q_2)$ for the queries equals to 1.

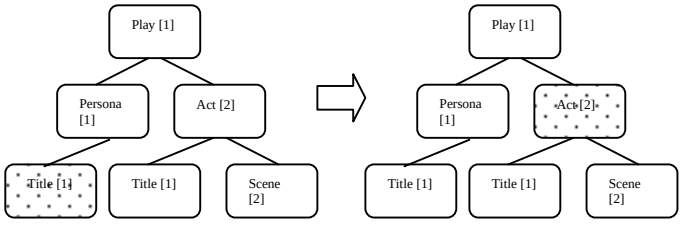

Fig. 5 Relationship G_4

(5) Cousin(G_5)

The Cousin() relationship, as shown in Fig. 6, represents the two nodes have a common ancestor and same depth. However, the queries should not have a common parent and the following criteria must be satisfied for Γ_5:

1. $D_V(Q_1, Q_2) = 0$
2. $Prefix(Q_1, Q_2) > 0$
3. $Prefix(Q_1, Q_2) < min\{Length(Q_2), Length(Q_1)\}$

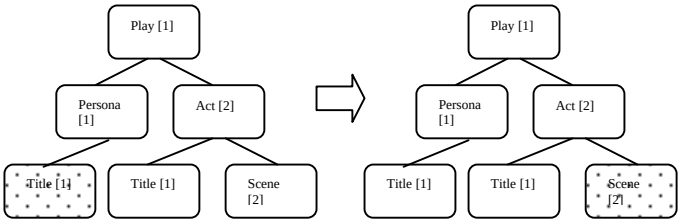

Fig. 6 Relationship G_5

3 Discovering Relationships

Although the usual support and confidence framework to assess association rules has several drawbacks[17], association rules mining are still widely used in prediction [5,8], user customizations [4] and profiling [7]. Particularly, we are interested in their uses in caching of XML query patterns [16]. XML documents structure mining can provide useful information for applications such as prefetching and rating of documents.

In this section, we describe a method that performs data mining on the patterns of XML queries with respect to the inter-query relationships discussed in Section 2.3. We aim to use the mining results as hints to predict future queries and user access patterns.

3.1 Steps of Relationship Mining

```
Obtain the query history for an XML document
Identify different user sessions in query log
TransSet = empty
For each user session {
        Result = empty
        (If the session is not empty) {
        For each sequential pair of queries {
            R = the inter-query relationship identified
            Result = Result ∪ R
        }
        TransSet = TransSet ∪ Result
}
```

Fig. 7 Preprocessing for Inter-Relationships Mining

In order to obtain the association rules for inter-query relationship. Firstly, we preprocess queries from the query log and transform them (a series of XPath expressions) into transactions as shown in Fig. 7. Suppose 'TranSet' represents the set of extracted transactions for user sessions, 'Result' denotes an atomic transaction for each session, and

'R' represents one inter-query relationship(item) for a sequential pair of queries. In the preprocessing step, we assume that each user session has issued a sequence of queries $Q_1Q_2Q_3Q_4Q_5\ldots Q_k$ against the same XML document. The corresponding sequence of relationship would be $R_1R_2R_3R_4\ldots R_{k-1}$ as shown in Fig. 8. Therefore, each user session is represented as a series of query relationships and forms an atomic transaction. With many user sessions, sequential mining algorithms can be applied .

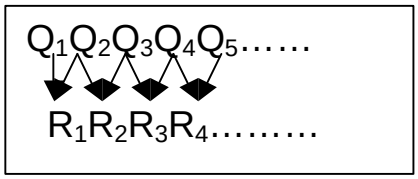

Fig. 8 Mapping relationships of XPath queries into a transaction

4 Applications of Inter-Query Relationships

Sequential association rules mined from the inter-query relationships provide the access patterns of the users in the past. With this information, various applications can take advantage of it when publishing large XML documents from RDBMS or ORDBMS.

4.1 The StructCache Framework

In [23], we have introduced the StructCache framework that takes advantage of the DTD in design of replacement algorithm, 'StructCache'. The 'StructCache' algorithm specifically handles XML objects in a different way than other web objects.

In this paper, we try to enhance the prefetching performance by incorporating the mined inter-query patterns obtained in section 3. Fig. 9 shows the revised internal structure of StructCache framework. In short, the 'Offline information' represents the identified query heuristics whilst 'Prefetching module' denotes the software module that adopts our suggested prefetching strategy. Hence, the revised architecture allows queries caching and prefetching. In next section, we will mention the details of the prefetching stragtegy.

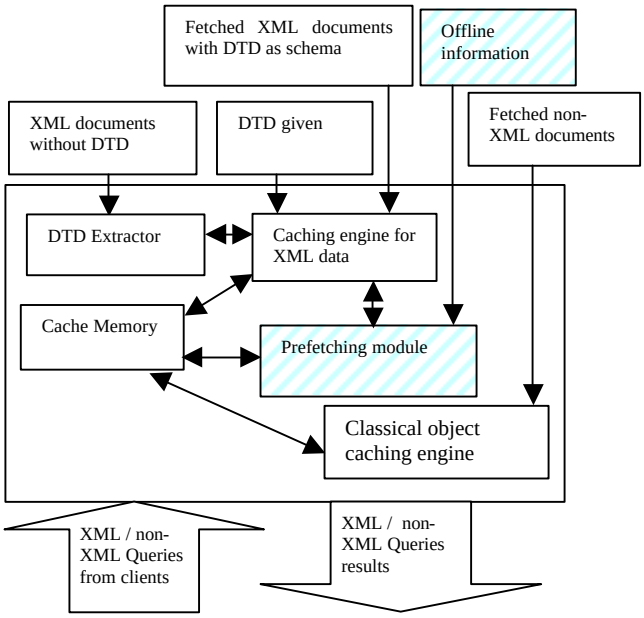

Fig. 9 Prefetching in revised StructCache framework

4.2 Prefetching of Web Objects

Mined query patterns can be used to support prefecthing of web objects. Fig. 10 shows the role of prefetching in queries answering in a user session. Let Q_{i+1} and Q_{i+1}' be the next and predicted data query respectively and R_{i+1} and R_{i+1}' denote the actual (if exists) and predicted relationship issued or to be issued by client respectively.

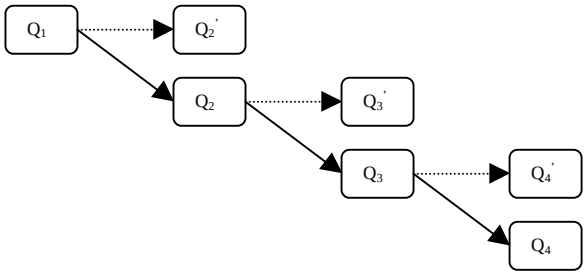

Fig. 10 Prefetching of queries using queries sequence

Rather than inferring Q_{i+1}' directly, we adopt another approach to infer the possible query by a predicted relationship. As shown in Fig. 11, predicted relationship R_{i+1}' is obtained from the sequential association rules of relationship sequence and actual queries sequence. The predicted results is set of possible queries, $\{Q_{i+1}'\}$. For our prediction, we utilize the last and second last executed query (Q_i , Q_{i-1}) instead of the whole query sequence in order to minimize the complexity.

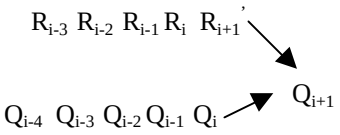

Fig. 11 Prefetching with relationship sequence

The details of query generation based on query relationships can be found in [22]. In [22] set relationships are defined between the constraints of antedent and consequent queries which are used by an pre-processing algorithm for generic rules extraction. Here, predicted result(s) of $\{Q_{i+1}'\}$ is/are prefetched to answer query Q_{i+1}. Assume R_i is the last derived relationship, each candidate Q_{i+1}' is a projection from function ⊗:

$$\{Q_{i+1}'\} = Ä(R_{i+1}', R_i, Dist(Q_i, Q_{i-1}), Q_i)$$

For each possible Q_{i+1}', there is an accompanied score (S_{i+1}') and is obtained from a scoring function (Ψ).

$$S_{i+1}' = Y(Q_{i+1}', R_{i+1}', R_i, Dist(Q_i, Q_{i-1}), Q_i)$$

where Ψ denotes a weighted combine function that takes the mined rules {A}, R_i, $Dist(Q_i, Q_{i-1})$ and Q_i into consideration.

If association rule mining is performed to discover the relationship patterns, we can develop the scoring function based on the candidate itemsets.

Suppose a, b, m ∈ [0,1] and $a > b$. **Support()** is a function returns the support count for an association rule whilst **ItemLength()** returns the item length of the candidate itemset.

Assume that an association rule of the form $a_{t} \Rightarrow a_j$ is used for the generation of Q_{i+1}' where $ItemLength(a_j \mid \forall a_k \exists a_j\ ItemLength(a_j) >= ItemLength(a_k) \wedge a_j, a_k \in \{A\})$ and $Support(a_j \mid \forall a_k \exists a_j\ Support(a_j) >= Support(a_k) \wedge a_j, a_k \in \{A\})$

Case 1: $Dist(Q_i, Q_{i-1}) <= 1$ and $R_i = G_1$

$S_{i+1}' = ItemLength(a_t) / ItemLength(a_j) * (m + Support(a_t) / Support(a_j))$

Case 2: $R_i = G_3$

$S_{i+1}' = ItemLength(a_t) / ItemLength(a_j) * (m + Support(a_t) / Support(a_j)) * (Dist(Q_i, Q_{i-1}))^{\alpha}$

Case 3: Others

$S_{i+1}' = ItemLength(a_t) / ItemLength(a_j) * (m + Support(a_t) / Support(a_j)) * (Dist(Q_i, Q_{i-1}))^{\beta}$

Note that the score is affected by support counts, antedent's item length, and distance function. By adjusting parameters a, b, m, we can avoid biasing to a single factor. Moreover, we are also able to obtain an optimal scoring function for a XML document by empirical results.

In this paper, we suggest 3 approaches to perform prefetching. The first approach is to select the predicted query/queries with top score. The second approach selects queries with scores higher than a given threshold. The last approach is to select the set of queries with scores higher than the score obtained with a reference association rule.

Let **Antedent()** be a function returns the antedent of an association rule and **i** denote the number of queries executed in a user session, prefetching is only done iff the following creteria are met:

1. $S_{i+1}' > 0$
2. $Antedent(a_t) \subseteq \forall t\ (\{R_i'\} \mid \{R_i'\} = \{R_{i-1}'\} \cup R_i \wedge 1 <= t <= i)$

The first criterion indicates that we would prefetch query if and only if there is at least one a_t such that the corresponding S_{i+1}' has non-zero value. The second criterion requires the antedent of a_t to appear in the possible enumeration of relationship sequence beginning from R_i'. Hence, length of each enumeration is different and the number of enumerations will be i if i >= 1 or 0 otherwise.

5 Conclusion

In this paper, we describe a framework to support prefetching of web objects through mining of inter-query relationships. We start with a presentation of a method to identify the possible relationships between XML queries. With a query log, sessions of enumerated query relationships can be formed. Through the mining of the relationships, we have developed a prefetching approach that makes use of the mined rules.

6 Acknowledgement

The work reported in this paper was partially supported by Hong Kong CERG Grant – PolyU 5094/00E.

References:

[1] W3C, *XML Path Language (XPath) 2.0*, http://www.w3.org/TR/xpath20, Dec 2001

[2] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases", *In Proceedings ACM SIGMOD Intl. Conf. Management of Data*, pages 207-216.

[3] Qiang Yang, Henry Haining Zhang, Ian Tian Yi Li, "Mining web logs for prediction models in WWW caching and prefetching", *KDD 2001*, 2001

[4] Andreas Geyer-Schulz and Michael Hahsler, "Evaluation of Recommender Algorithms for an Internet Information Broker based on Simple Assocation-Rules and on Repeat-Buying Theory", *Proceedings of the WebKDD 2002*, 2002

[5] E. Frias-Martinez and V. Karamcheti, "A Prediction Model for User Access Sequences", *Proceedings of the WEBKDD Workshop: Web Mining for Usage Patterns and User Profiles, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 2002

[6] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. "Data Preparation for Mining World Wide Web Browsing Patterns", *Journal of Knowledge and Information System 15-332*.

[7] F. Abbattista, M. Degemmis, N. Fanizzi, O. Licchelli, P. Lopes, G. Semeraro, F. Zambetta, "Learning User Profiles for Content-Based Filtering in e-Commerce", *Proceedings Convegno Associazione Italiana per Intelligenze Artificale(AIIA 2002)*, Sep 2002

[8] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos, "Effective Prediction of Web-user Access : A Data Mining Approach", *In Proc. of Workshop on WebKDD 2001*, San Francisco, CA, Aug 2001

[9] Chen, L. and Rundensteiner, EA "ACE-XQ: A Cache-aware Xquery Answering System", *ACM SIGMOD Associated Workshop on the Web and Databases(WebDB)*, Madison, Wisconsin, June 2002

[10] Chen L., Wang. S. and Rundensteiner, EA "A Fined-Granied Replacement Strategy for XML Query Cache", *ACM International Workshop on Web Information and Data Management(WIDM)*, McLean, Virginia, November 2002

[11] Carlo Sartiani, "A Framework for Estimating XML Query Cardinality", *WebDB 2003*, Jun 2003

[12] Dongwon Lee and, Wesley W. Chu, "Semantic Caching via Query Matching for Web Sources", *Proc. ACM CIKM Intl Conference on Information and Knowledge Management*, Nov 2-6, 1999

[13] Dan Duchamp, "Prefetching Hyperlinks. USENIX Symposium on Internet Technologies and Systems", *AT&T Labs*, 1999

[14] L. Fan, P. Cao, W. Lin, and Q. Jacobson, "Web prefetching between low-bandwidth clients and proxies: Potential and performance", In Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '99), Atlanta, GA, May 1999

[15] Edith Cohen, Haim Kaplan, "Prefetching the Means for Document Transfer: A New Approach for Reducing Web Latency", *INFOCOM 2*, 2002

[16] Liang Huai Yang, Mong Li Lee, Wynne Hsu, "Efficient Mining of XML Query Patterns for Caching", School of Computing National University of Singapore VLDB-03, 2003

[17] Fernando Berzal, F., Blanco, I., Sánchez, D., & Vila, M., "A new framework to assess association rules", *In F. Hoffmann (Ed.), Advances in intelligent data analysis. Fourth international symposium IDA'01*, 2001

[18] P. Ciaccia and W. Penzo, "Adding Flexibility to Structure Similarity Queries on XML Data", *In Proceedings of the 5th International Conference on Flexible Query Answering Systems (FQAS 2002)*, Copenhagen, Denmark, October 2002

[19] Laurentiu Cristofor, "ARMiner Project", *UMass/Boston*, Spring 2000

[20] W. Penzo, "Integration of Semantic and Structure Similarity for XML Data Ranking", *In Atti del Decimo Convegno Nazionale su Sistemi Evoluti per Basi di Dati (SEBD 2002)*, Portoferraio, Italy, June 2002

[21] P. Ciaccia and W. Penzo, "The Collection Index to Support Complex Approximate Queries on XML Documents", *In Proceedings of the First International XML Database Symposium (XSym03)*, Berlin, Germany, September 2003

[22] Vincent Ng and Chi Kong Chan, “Mining Patterns for Prefetching XML Queries”, *Department of Computing, The Hong Kong Polytechnic University*, 2004

[23] Shiu Hin Wang and Vincent Ng, “Structural Caching XML data for Wireless Accesses”, *In Workshop on Applications, Products and Services of Web-based Support Systems*, Halifax, Canada, October 2003