Relaxing XML Preference Queries for Cooperative Retrieval

SungRan Cho and Wolf-Tilo Balke,

L3S Research Center, Leibniz University of Hannover, 30167 Hannover, Germany
{scho, balke}@L3S.de

Abstract. Today XML is an essential technology for knowledge management within enterprises and dissemination of data over the Web. Therefore the efficient evaluation of XML queries has been thoroughly researched. But given the ever growing amount of information available in different sources, also querying becomes more complex. In contrast to simple exact match retrieval, approximate matches become far more appropriate over collections of complex XML documents. Only recently approximate XML query processing has been proposed where structure and value are subject to necessary relaxations. All the possible query relaxations determined by the user's preferences are generated in a way that predicates are progressively relaxed until a suitable set of best possible results is retrieved. In this paper we present a novel framework for developing preference relaxations to the query permitting additional flexibility in order to fulfil a user's wishes. We also design IPX, an interface for XML preference query processing, that enables users to express and formulate complex user preferences, and provides a first solution for the aspects of XML preference query processing that allow preference querying and returning ranked answers.

Keywords: XML query processing, preference-based retrieval, personalization.

1 Introduction

XML is widely used as a base technology for knowledge management within enterprises and dissemination of data on the Web (like e.g., product catalogues), because it allows to organize and handle semistructured data. A collection of XML documents is viewed as a forest of node labeled trees. The data generally can be queried on both structure and content using advanced retrieval languages such as XPath or XQuery. User queries will usually be structured to express the user's information needs and users often have quite specific preferences about the structure, especially when the document structure shows a certain semantics often described by DTDs or XML schema. However, due to the large number and complexity (or heterogeneity) of XML documents, the retrieval process should be cooperative between system and user. Here approximate matches that allow to rank answers according to their relevance to the query [3, 6, 19, 20] are more appropriate than exact match queries. Recently several proposals have therfore studied ranking methods that account for structure to score answers to XML queries.

When querying for information users may only have a vague idea of what type of documents can be expected in the collection as well as where the information might occur in a specific XML document. Personal preferences are a powerful means to express user wishes that might not always be fulfilled, but allow for relaxation of query predicates by user provided alternatives to search desired information and order the search results. Hence, preferences combat two undesirable scenarios: to return an empty result and to flood the user with too many results. Relaxation of overspecified queries avoids empty results and to cope with flooding a pruning of less relevant answers can be performed (top-k querying).

In this paper we focus on the *relaxation of structural preferences* in a query, inspired by fair query relaxation techniques applied to the query structure [2]. In order to avoid empty results and to further personalize user queries, a preference query considers node relaxations to the preferred query structure to return 'closest' (or the most relevant) results to the user request. Preference queries are rewritten into a set of queries progressively posed to the database (unfolding): starting with a highly specific query with all top attributes from a user's preferences, each predicate is gradually relaxed to less preferred attributes (*base relaxation*). Moreover, preferred structures in the query can be relaxed to all still relevant query structure (*secondary relaxation*).

Since preferences generally induce a certain ranking on the result set, an unfolding sequence can be enumerated by successively relaxing query predicates from most to least preferred attributes in each preference, considering all variations as query rewritings, and collecting the results with respective rank information. In this paper we additionally define an ordering scheme that provides a fair relaxation by taking into account the order induced by not only multiple preference attributes (structure and value), but also their relaxed versions. Efficient preference XML query processing now requires the ability to execute a set of relevant queries and determine most relevant results as induced by user provided preferences. We design IPX, an interface system for effective XML preference query processing. IPX incorporates XML query processing enhanced with preference features providing not only a extension of XPath syntax, but also dynamic preference operations such as an ordering method and top-k processing.

2 XML Preference Queries

For a complete overview of our structure-based XML preference framework see [7]. We consider a data model where information is represented as a forest of node labeled ordered trees. Each non-leaf node in the tree has a type as its label, where types are organized in a simple inheritance hierarchy. Each leaf node has a string value as its label. Simple data instances are given in Figure 1.

Beyond simple lookups modern database retrieval enables users to express certain preferences. Such preferences state what a user likes/dislikes and preference query processing attempts to retrieve all best possible matches in a cooperative fashion avoiding empty result sets. In relational databases such preferences are specified over data values. But, since in XML retrieval also the structure of the document often carries semantics described by a DTD or XML schema, user preferences can also be

specified over the structure. Qualitative preferences are often expressed by partial orders and visualized by a graph where each node is labeled by values of a particular type and the direction of each edge between nodes expresses that the label of the node where the edge originates, is preferred over the label to which the edge points. The example structural preference of Figure 2(b), SP_{treat}, for instance, expresses that *toy* is preferred over *hair-care*.

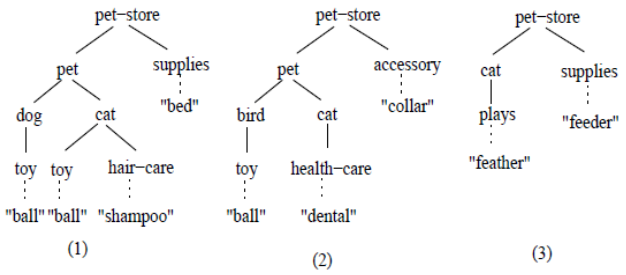

Fig. 1. Example for heterogeneous XML databases.

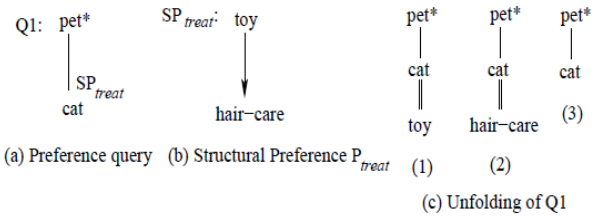

Fig. 2. Example query and structural preference.

Like XML documents, queries are usually structured to express the users' information needs. All existing query languages for XML are tree-shaped, where nodes are labeled by types or string values, and edges correspond to parent-child or ancestor-descendant relationships. Figure 2 (a) shows a simple path query with a preference: each edge represents a parent-child relationship (a double edge is an ancestor-descendant relationship). Some nodes are marked with '*', indicating that they are projected (i.e. returned as answer). Any leaf nodes can be marked with explicit preferences.

A preference query then needs to be rewritten as an ordered set of queries induced by the explicit preference (referred to as query process), retrieving all data relevant to answering the original query. We thus have to examine all possible relaxations of the user preferences and expand the preference-marked nodes accordingly. Next we define the unfolding of preference queries for query processing in case of a single preference in a query:

Definition 2.1. [*Unfolding Preference Queries*] Given a query Q with a node n marked with a preference P, an expanded query Q' is obtained by unfolding Q:
(i) while respecting the order induced by P adding a node v of P to a query node n as a successor (in the case where n is value-constrained, v resides between n and the value), adding an edge with ancestor-descendent label from n to v, and propagating a possible distinguished status of n down to v,
(ii) entirely removing P from Q. ■

This results in an ordered set of $|P + 1|$ distinct queries: the most preferred query is expanded by (one of) the most preferred node(s) in the preference graph and the least preferred query of the expansions is obtained by simply removing mark P from query node n without adding any new nodes and edges. An incoming edge of a preference node in a query is generalized to an ancestor-descendant relationship allowing all matchings to relevant portions of the XML database.

Given query Q_1 in Figure 2(a) containing a structural preference SP_{treat}, the induced order providing a desirable stepwise relaxation scheme is reflected by all possible expansions of Q_1 in Figure 2 (1), (2), and (3) in the unfolding process. Starting with query (1), whose answer set the user prefers most over queries (2) and (3), and down to query (3) as the least preferred query. With only a single preference considered, we encounter 3 different rewritings for the query in the unfolding process.

Generally speaking, preference attributes whether they are value or structural information, are progressively relaxed, i.e., starting with all top attributes stated in a user's preference and gradually relaxing to less preferred attributes. To capture preferred answers to a given preference query we first generate the most specific relaxed preference queries on a set of queries:

Definition 2.2. [*Base Relaxation*] Let Q be an XML preference query. Then query Q' expanded with preference nodes from P in the unfolding process is referred to as base relaxation. ■

For example, given the query 2, the queries generated from the base relaxations are Figure 2(c) (1) and (2) (but not (3)). Our focus is closely related to the issue of loss of desirability, i.e. how much a preference query is relaxed. To address this problem we next propose a framework for relaxing structural preferences in an organized way.

3 Relaxing Queries based on Structural Patterns

In this section we formally define approximate preferred queries based on the notion of structural pattern relaxations. As a base, we take the node generalization relaxation defined in [2], and generalize it in the context of preferences. In particular, we consider three specific preference relaxations that all are structural relaxations of the preference query. A set of predefined base preferences is provided by each user, all of which can be described as partial order graphs and alternative sets of preferences are generated with respect to a notion of structural similarity. We consider relaxing preference nodes in the query, which we refer to as secondary relaxation. Such secondary relaxation is closely related with similarity searches. While usually an ontology plays a central role in a similarity search, here we simply use DTD or XML Schema information to promote the relevant relaxed nodes.

Node Generalization: *This permits the type of each preference node to be generalized to a supertype. For example consider the expansion query of Figure 2(c) (1) and a type hierarchy containing the path "toy|plays|necessity". The toy can be generalized to plays followed by necessity, allowing for arbitrary play and pet necessities to be returned instead of just toy information. This kind of node relaxation is appropriate, whenever no exact match is found.*

Sibling Promotion: *This permits the corresponding sibling DTD nodes of each preference node to be promoted. For example, if the query of Figure 2(c) (1) does not result in a match, but there may be a lot of cat related products that come very close to a toy. Thus, this near preferred sibling node could be relevant, if no exact match is available.*

Path Node Promotion*: This also uses DTD information to promote relevant relaxed nodes. It permits the nodes to be promoted on the corresponding DTD path between a query node and an expanded preference node (excluding the original query and preference nodes themselves). For example, consider the query in Figure 2(c) (1). If there is a node product on the DTD path between the corresponding cat and toy, in the near node product can be promoted as an alternative closest node of the toy node.*

Preference queries have to be relaxed into a set of queries that is guaranteed to retrieve every possibly relevant document from an XML database. In the relaxations above, only preference nodes in the query are relaxed and approximate matches in the corresponding relaxed nodes are retrieved. These relaxations do not increase the query size, but increase the set of those candidate queries close to the original query.

So far, we have focused on generalizations of preference nodes while keeping a generalized edge of ancestor-descendant type. However, users might wish to retrieve preferred answers associated with some edge information. In order to permit an additional specification in preferences, we allow users to specify if edges should be constrained with respect to their depth.

Edge Depth Preference: *This permits a depth range of the appended preference node in order to limit the search radius. For example, in the query 2(c) (1), the user could constrain the edge (cat, toy) with maximum depth information, retrieving only toys that are within a certain distance from the cats.*

4 Ordering Relaxations

An ordering method is necessary to distinguish different relaxations of the initial user query and get a notion of what is a *minimum amount of relaxation*. For ranking, several techniques have been proposed, like e.g., approximate keyword queries based on ontologies [22], or the *tf*idf* measure of the IR community that matches keyword queries against a document collection. In any case, the general approach to relaxation is in line with our development in the paper.

Our base relaxation ordering method uses Pareto optimality to rank all combinations of preference nodes and their relaxed versions expanded in the query. But the challenge in this paper is also to organize a framework for ordering a total relaxation of a preference query, i.e. including secondary relaxations. We will now define an explicit relaxation order that distinguishes the loosened preference nodes.

Definition 4.1. [*Relaxation Order*] Let Q be an XML preference query and Q' be an expansion query. Then the preference node is relaxed in the following sequence:

1. *node generalization*: replaces its immediate supertype of the preference node recursively;
2. *sibling promotion*: replace its corresponding DTD sibling nodes of the preference node;
3. *path node promotion*: replaces the parent of the preference node in the corresponding DTD path recursively;
4. *preference node deletion*: finally deletes the preference node. ■

While Definition 4.1 (2) takes into account the local closeness that treats all siblings of a given preference node equally, Definition 4.1 (1), (3), and (4) account for the subsumed closeness of the corresponding preference node. Intuitively, node generalization is considered as the most specific (or most precise) relaxation, since subsuming nodes are very closely related to the preferred node. The sibling promotion is next, because it leads to a less specific, but often still relevant relaxation, and the DTD path node promotion is last, because it is a generalization of sibling promotion.

The sequence of our relaxation can quantify the closeness of an answer in the collection of documents. In our framework, the ranking method is monotonic since each relaxation step always follows the dominance relation and the relaxed query includes all the nodes from the original query, i.e., only relaxing expanded nodes. For example, given the query Q_1 in Figure 2(a), Figure 3 shows the sequence of different relaxation to the preference nodes *toy* and *hair-care*.

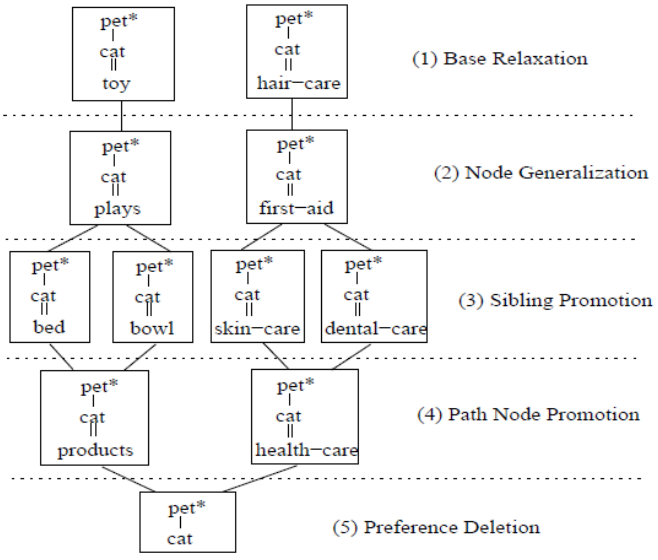

Fig. 3. Example preference relaxation process.

In our basic relaxation ordering scheme, we still need to distinguish the degree of subsumed closeness in order to increase precision of ranking. The general rationale

for this closeness is to use the distance on the DTD path or the path in the type hierarchy between the original query node and the relaxed preference node, i.e., if the distance gets larger, the degree of closeness to the initial query decreases. The distance measure becomes useful to compute the score (or rank) of answers to decide how closely they match the query. If a query node contains a single preference P, where P consists of $|P|$ distinct attributes, the total number of relaxed queries from the secondary relaxation is:

$$\sum_{i=1}^{|P|} |Sibling_i| + |SuperType_i| + |DTDpathNode_i|$$

The generalization of definition 4.1 to the case of multiple nodes marked with preferences is straightforward: each single preferred node is relaxed such that every possible combination generated from base and secondary relaxations is reflected by a relaxed query.

5 Dealing with Multiple Preferences

In this section, we discuss incorporating multiple preferences, especially in the case where a query node is marked with multiple preferences. The unfolding process of the query should be based on the semantics of the query we consider. This is a generalization of definition 2.1. If query nodes are marked with a single preference, the total number of queries relaxed from the base relaxation is $|P_1+1| \times \ldots \times |P_n+1|$, where each preference P_i consists of $|P_i|$ attributes.

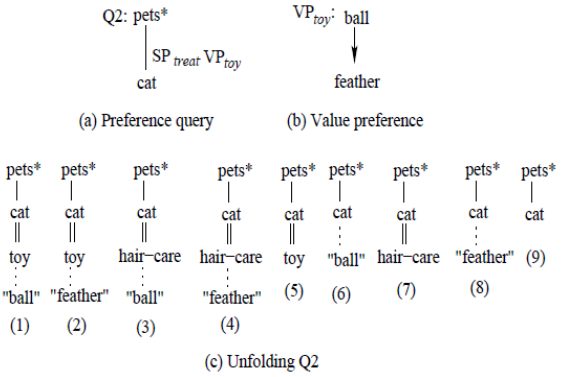

Fig. 4. Example query with structural and value preferences.

Individual users may have specific structural and value preferences. For evaluation it is necessary to combine different preferences specified on a single query node. To explain the basic idea, we consider the case where query nodes are marked with a single structure preference and a single value preference together in this subsection.

If a query node is marked with a single value preference, an unfolding of the query is the same as we have shown in Figure 2. However, in the case where a query node can be marked with structural and value preferences together, we need to expand the query with a set of queries in a way to comply with the semantics of query described in Section 2. Thus, structural elements should be expanded prior to values because in XML data and queries only leaf nodes specify their values.

For example, consider the query Q_2 in Figure 4(a), where Q_2 contains the structural preference SP_{treat} in Figure 2(a) and the value preference VP_{toy} in Figure 4(b). The query Q_2 is rewritten into a set of queries in Figure 4(c) by expanding structural elements followed by values. The query Q_2 produces 9 possible expansions in Figure 4(1)-(9) in the unfolding process.

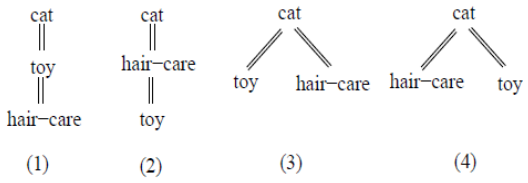

Fig. 5. Example of multiple structural preferences.

Next we discuss additional flexibility of structural preferences in the query. Multiple preferences can be specified on a single query node as well as on multiple query nodes in the query. For example, consider a query consisting of *cat* marked with two structural preferences (for ease of understanding we will use two times SP_{treat}). We now need to consider all possible query structures encountered. Figure 5 shows the relaxed queries. A key part of defining a relaxation order is to examine all possible structure combinations in the query, which is shown in the query in Figures 5(1)-(4). In particular the queries in Figures 5(3) and (4) are valid, especially when the order of queries is material. However if the order of queries is not concerned, Figures 5(3) and (4) are equivalent because they return the same answers. However by referencing the DTD, the number of relaxed queries encountered in the base and secondary relaxation processes may be reduced. For preferences on element tags, the respective DTD can help to prune a set of relaxed queries by simply testing if the relaxed node is valid in the query structure.

6 Design of IPX

For evaluating our concepts, we designed an interface called IPX to enhance queries with preference operations. Since preferences are specified on top of query expressions in languages such as XPath and XQuery, IPX can be implemented on top of commercial XML servers supporting such queries. Figure 6 shows the overall architecture of IPX. IPX is mainly composed of three components: query rewriter, preference handler, and ranking handler. The IPX architecture has successfully been demonstrated at ACM SAC 2009, see [8].

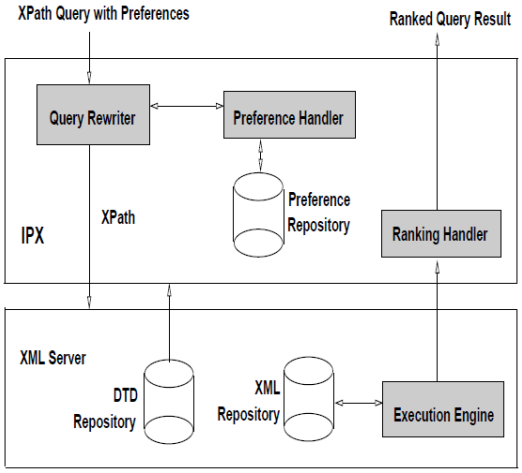

Fig. 6. Overall architecture.

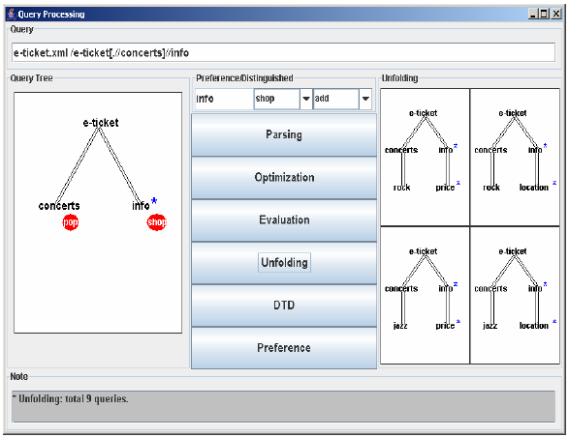

Fig. 7. IPX Preference XML query processor.

The IPX first accepts a user query containing structural and/or value preferences. Then the *query rewriter* rewrites the query into conventional XPath or XQuery queries which can be executed in any conventional XML engine, by expanding the query with the user provided preferences. In particular, if a query contains structural preferences, the rewriter checks the corresponding DTD to expand relevant elements. Since IPX handles ordered answers, the *ranking handler* determines the necessary set of queries and preserves the induced order of the result set by quantifying the relevance of the relaxed queries. In addition, since structural and value preferences of individual users need to be maintained, the *preference handler* parses them and stores them in a repository. It also is designed to manage multiple granularities of preferences to inte-

ract with multiple preference functions given in the query, and to compute their conjunction currently following the Pareto semantics (of course our framework is open for extensions). Moreover, it interacts with the query rewriter to support the unfolding process. Furthermore IPX encompasses the following functionality and features:

Extending XPath syntax: IPX implements several flexible syntaxes, which incorporate preference assignment to the query and necessary preference operations.

Supporting top-k processing: IPX also supports the efficient evaluation of top-k queries that retrieve only the k best answers. The *preference handler* and *ranking handler* allow to synchronize the top-k retrieval.

Query optimizer: Since a preference has to be unfolded into a set of queries and such expansions typically contain redundancies, it is important to identify and simplify necessary relaxed queries for effective evaluation. IPX implements a *preference query optimizer* that not only determines an optimal set of expansion queries, but also preserves an ordering induced by the preference.

Improving query evaluation: IPX implements evaluation techniques to improve evaluation times for a preference query by considering the special features of preference queries that typically contain repetitive fragments and always follow induced patterns.

Visualization: IPX implements a flexible and interactive graphical interface that facilitates browsing of different preference graphs and DTDs, and displays user queries as well as the results of query evaluations. For example, given the query shown in the left window of Figure 7, some queries in the unfolding process are displayed in the right window of Figure 7, where the first relaxed query is the most preferred query, the second and the third queries are next and etc.

7 Related Work

Preferences recently are an active research area in information systems research [9, 1, 14, 15, 21, 13]. Due to the importance in practical applications, preference query processing has attracted considerable attention leading to a large number of possible techniques.

Several systems for supporting preference queries have been recently proposed. In [10], the author investigated the semantic optimization of preference queries to remove redundant occurrences of preferred values in relational databases. Recent work by [15] proposed preference XML queries in connection with the modeling framework for preferences as partial order graphs given in [14]. The resulting language enables the use of soft filtering conditions in contrast to conventional exact match conditions for node-selection in XPath. Like in our approach a soft condition defines a strict partial order over the set of elements to be filtered and then returns only the best matches. In [4] authors proposed a fair scheme of relaxing preference values to pose to the database such that the user retrieves data in a well-defined order and can choose at any stage, whether he is already satisfied by the result and the query processing can be terminated. Another interesting study is presentational preferences for XML query results. Although in contrast to our approach these preferences do not affect the matching process, some basic techniques exploiting the DTD structure are

related. Since XQuery is a data-transformation query language, users can easily define a set of preferences to impose an ordering on the presentation of the results.

Another related area of work is scoring. In particular scoring for XML databases has actively been studied [2, 5, 6, 19, 20]. While our approach gives a fair relaxation method, these approaches promote traditional scoring methods for XML such as probability-based approaches, considering path expressions along with query keywords, and incorporating a similarity measure. Recognizing an exact match retrieval model as often inadequate in practical scenarios, the resulting XML query engines (e.g., XIRQL [11] and XXL [22]) have already been extended to support an IR-style matching of data values. Generally these engines allow finding similar values for a certain predicate or relaxing query predicates to find desired values in related structural elements.

There are also several XML query relaxation proposals (see, e.g., [2, 3]). In particular, [2] addressed the problem of approximate XML query matching based on tree pattern query relaxations and provided efficient algorithms to prune query answers that will never meet a given threshold. While before the focus was on defining a framework for structural relaxation, the work in [3] focuses on scoring methods on both structure and content to evaluate top-k answers to XML queries in the same relaxation framework. Building on previous approaches for preferences in information systems, in this paper we presented a framework for how XML preference queries can be relaxed not only by such preference information, but also by similarity of given preferred information.

8 Conclusions

In this paper we presented a framework for relaxing and combining structural preferences to search personalized information on XML databases or semi-structured document collections like e.g., enterprise document collections or e-catalogues.

In order to fulfill the user's interest to the best possible degree, we considered preference query relaxations to retrieve closest relevant results to a user request. We showed that our framework can be applied to existing XPath engines by a syntactic enhancement that incorporates function calls in the query. We also designed and implemented IPX, a flexible interface for handling preferences in the query. IPX is extensible to incorporate other necessary applications such as automatic extractions of structural preferences using DTDs and/or user profiles. IPX thus can help sharing efforts when developing preference XML engines.

We are currently investigating extensions to IPX to support conventional scoring methods which account for both structure and value. At the same time we also address the problem of how to score answers for complex joint matches. In order to improve efficiency of the overall preference query processing, we see many interesting directions of future work such as efficient preference query evaluation and optimization strategies. Solutions to these problems will contribute to the application of preference-based personalization in XML queries.

References

1. Agrawal, R., Wimmers, E., 2000. A framework for ex-pressing and combining preferences. In ACM SIGMOD Conference on Management of Data, Dallas, TX, USA.
2. Amer-Yahia, S., Cho, S., Srivastava, D., 2002. Tree Pattern Relaxation. In Int. Conference on Extending Database Technology (EDBT), Prague, Czech Republic.
3. Amer-Yahia, S., Koudas, N., Marian, A., Srivastava, D., Toman, D., 2005. Structure and content scoring for XML. In Int. Conference on Very Large Databases (VLDB), Trondheim, Norway.
4. Balke, W., Wagner, M., 2004. Through different eyes: assessing multiple conceptual views for querying Web services. In World Wide Web Conference (WWW), New York, NY, USA.
5. Bremer, J., Gertz, M., 2002. XQuery/IR: Integrating XML document and data retrieval. In WebDB, Madison, WI, USA.
6. Chinenyanga, T., Kushmerick, N., 2001. Expressive and efficient ranked querying of XML data. In WebDB, Santa Barbara, CA, USA.
7. Cho, S., Balke, W., 2008. Order-Preserving Optimization of Twig Queries with Structural Preferences. In IDEAS, Coimbra, Portugal.
8. Cho, S., Balke, W., 2009. Building an Efficient Preference XML Query Processor, In ACM Symposium on Applied Computing (SAC), Honolulu, HI, USA.
9. Chomicki, J., 2002. Querying with intrinsic preferences. In Int. Conference on Extending Database Technology (EDBT), Prague, Czech Republic.
10. Chomicki, J., 2004. Semantic optimization of preference queries. In Int. Symposium on Applications of Con-straint Databases, Paris, France.
11. Fuhr, N., Großjohann, K., 2001. XIRQL: A query language for information retrieval in XML Documents. In ACM SIGIR, New Orleans, LA, USA.
12. Guo, L., Shao, F., Botev, C. Shanmugasundaram, J., 2003. XRANK: Ranked keyword search over XML document. In ACM SIGMOD Conference on Management of Data, San Diego, CA, USA.
13. Kanza, Y., Sagiv, Y., 2001. Flexible queries over semi-structured Data. In Int. Symposium on Principles of Database Systems (PODS), Santa Barbara, CA, USA.
14. Kießling, W., 1999. Foundations of preferences in database systems. In Int. Conference on Very Large Databases (VLDB), Hong Kong, China.
15. Kießling, W., Hafenrichter, B., Fischer, S., Holland, S., 2001. Preference XPATH: a query language for E-commerce. In Konferenz für Wirtschaftsinformatik Augsburg, Germany.
16. Koch, C., Scherzinger, S., Schweikardt, N., Stegmaier, B., 2004. FluXQuery: an optimizing XQuery processor for streaming XML data. In Int. Conference on Very Large Databases (VLDB), Toronto, Canada.
17. Papadias, D., Tao, Y., Fu, G., Seeger, B., 2003. An optimal and progressive algorithm for skyline queries. In ACM SIGMOD Conference on Management of Data, San Diego, CA, USA.
18. Papakonstantinou, Y., Vassalos, V., 1999. Query rewriting for semi-structured data. In ACM SIGMOD Confe-rence on Management of Data, Philadelphia, PA, USA.
19. Polyzotis, N., Garofalakis, M., Ioannidis, Y., 2004. Approximate XML Query Answers. In ACM SIGMOD Conference on Management of Data, Paris, France.
20. Schlieder, T., 2002. Schema-driven evaluation of approximate tree-pattern queries. In Int. Conference on Ex-tending Database Technology (EDBT), Prague, Czech Republic.
21. Stolze, M., Rjaibi, W., 2001. Towards scalable scoring for preference-based item recommendation. In Bulletin of the IEEE Technical Committee on Data Engineering.
22. Theobald, A., Weikum, G., 2002. The index-based XXL search engine for querying XML Data with relevance ranking. In Int. Conference on Extending Database Technology (EDBT), Prague, Czech Republic.