Building an Efficient Preference XML Query Processor

SungRan Cho
L3S Research Center
University of Hannover, Germany
scho@L3S.de

Wolf-Tilo Balke
L3S Research Center
University of Hannover, Germany
balke@L3S.de

ABSTRACT
Today user-centered information acquisition over collections of complex XML documents is increasingly in demand. To this end, preferences have become an important paradigm enabling users to express individual interests and delivering personalized information. As the structure of XML documents plays a major part in retrieval, users often have specific preferences about the structure. For evaluation a query has to be unfolded into an entire set of queries filling the structure with more or less preferred values. Since such structure expansions typically contain redundancies, it is important to identify and simplify necessary expansion queries for effective evaluation. To address these issues, we developed a preference query optimizer that not only determines an optimal set of expansion queries, but also preserves the specific ordering induced by the user preferences with respect to Pareto optimality.

Categories and Subject Descriptors
H.3.3 [**Information Systems**]: Information Search and Retrieval – *query formulation, retrieval models, search process.*

General Terms
Algorithms, Performance.

Keywords
XML databases, preference-based retrieval, personalization.

1. INTRODUCTION
Today XML is ubiquitous in retrieving and exchanging information over the Internet. All kinds of structured information can be expressed in XML documents and subsequently queried using advanced retrieval languages like XPath or XQuery taking into account both: the values of data items and the structure of the document they are found in. Like the XML documents, queries are usually structured to express the users' information needs. Especially, if the document structure shows a certain semantics (often described by a DTD), evaluating queries with such *structural preferences* is necessary. Human preferences have recently gotten considerable attention (see, e.g., [3, 4]), focusing on issues such as how to model preference queries in databases using partially ordered graphs, or evaluation algorithms, where query processing can be stopped after one or more optimal matches with respect to Pareto optimality have been found.

Actually, when querying for information users usually have a vague idea of what kind of information they look for, as well as where it should occur in a specific XML document. Preferences are hence becoming an important paradigm in query processing that allow for personalization respecting the users' interests. But up to now preference queries did only consider *values*, whereas in XML queries also preferences on *element tags* are valid, i.e., preferred tags become part of the query structure. For evaluation a preference XML query is rewritten into a *set of queries* progressively posed to the database: starting with a query expanded with all top attributes as stated in a user's preferences, each predicate is gradually relaxed to some less preferred attributes, until finally the most general query without any preference attribute is posed.

In this paper, we present our prototypical system for processing XML queries containing preference information. In particular, our system implements the expansion of XML queries by preference information together with a complete optimization framework for structural preferences. Our sophisticated optimization algorithm (as described in detail in [2]) allows to efficiently determine an optimal set of order preserving expansion queries, which can subsequently be evaluated by our system.

2. ARCHITECHTURAL ISSUES
Our framework is responsible for efficiently evaluating preferences expressed by individual users with respect to a DTD. Typically preferences are described as *partial order graphs*. Then an *XPath query* with structural preferences is parsed, optimized, and executed. The architecture of our system is depicted in Figure 1.

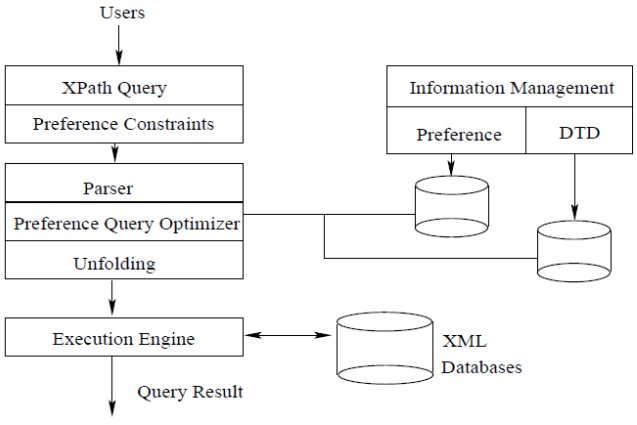

Figure 1. Architecture of the Query Processor

SAC'09, March 8-12, 2009, Honolulu, Hawaii, U.S.A.

Basically our preference XML query processor consists of two subsystems: the preference query processor and the information management system. The first system implements an XML query processor enhanced with preference operations that accepts user queries and preferences, and then parses, optimizes, and executes the queries. The second system incorporates the specification and management of the DTD and all preference information. The system encompasses the following functionality and features:

Query Interface. As a starting point users can formulate XPath queries for evaluation. Besides hard query constrains on structural elements or values (which are evaluated like in conventional XPath engines and thus serve as result filters), users can also augment structural elements of the query with preferences to express their personal information needs.

Parser. Once a user query is issued, the query is displayed as a labeled tree where nodes are labeled by an element tag, edges are labeled as parent-child (a single line) or ancestor-descendant (a double line), and one node is marked by '*' corresponding to the elements returned as query result. All preference information at query nodes can be dynamically assigned on demand through a graphical interface. Currently, in our prototypical system only leaf nodes can be marked with preferences.

Preference Query Optimizer. We use the term expanded node to denote a preference attribute in a query. A query marked with structural preferences is *unfolded* by additionally expanding the query node(s) with attributes in the given preference graph(s) before evaluation. Edges of expanded query nodes are generalized to ancestor-descendant relationships that match (relaxed) relevant portions of XML databases. When preference-marked nodes are distinguished nodes, their expanded nodes inherit the status of projection. This is because answers have to contain the projected nodes, but should also show in how far the users' preferred nodes could be granted. If some of expansion queries retrieve nested or identical results, not all queries have to be posed against the data collection. Analyzing all possible queries in the unfolding, the optimizer implements three distinct analysis techniques that have been described and evaluated in detail in [2]. In any case the preference query optimizer determines an optimal set of expansion queries, which in turn are minimal in their expanded nodes.

Query Unfolding Engine. This component focuses on the process of unfolding individual expansion queries before evaluation. A user can view optimized expansions of a query and interact with the results. For example, given the query shown in the query tree window in Figure 2, only two queries survived the optimization. Both optimized surviving queries are displayed in the right-hand side windows of Figure 2, where the first expansion is the most preferred query and the second query is less preferred. A user now can decide whether all queries should be posed, or just a specific subset, etc. Users can also specify a maximum size of the result set like e.g., the best ten documents.

Execution Engine: Once an optimal set of expansion queries has been generated, all individual queries are sent to the execution engine respecting the sequence of preference. This component executes each query and returns its answers. Since overlapping parts of all queries in the sequence have been removed the individual answer sets can simply be concatenated to reflect the final ranking of results. If the user has specified a maximum answer set size (*top-k querying*), the evaluation of the query sequence is stopped after the number of *k* results has been reached.

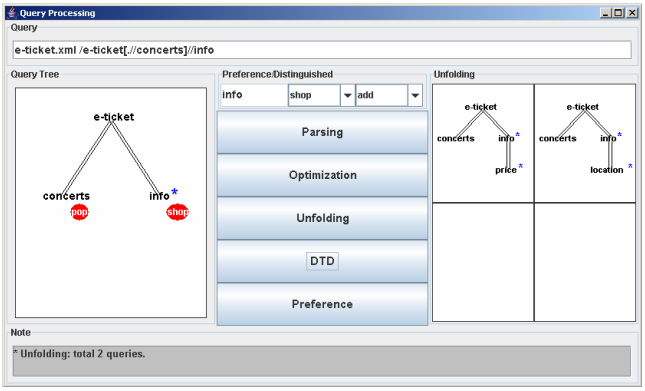

Figure 2. Screenshot of the Query Interface

3. CONCLUSIONS

In this paper we presented the architecture and the prototypical implementation of a preference XPath query processor. In particular, our system focuses on an order-preserving query optimization that determines not only an optimal set of expansion queries, but also an optimal set of expanded nodes in individual expansion queries. Moreover, it facilitates dynamic assignment of preference information to query nodes, as well as displaying optimized expansion queries after the unfolding process.

Currently, we are investigating extensions to our query processor incorporating more sophisticated evaluation techniques like for instance caching and reusing the results of structural joins within the query sequence, to further improve the efficiency of preference-based retrieval in XML databases.

4. ACKNOWLEDGMENTS

This work was funded within the Emmy-Noether Program of Excellence by the German Research Foundation (DFG).

5. REFERENCES

[1] Amer-Yahia, S., Cho, S., Lakshmanan, L. and Srivastava, D. Minimization of tree pattern queries. In *Proc. of the ACM SIGMOD Conf.*, Madison, WI, USA, 2002.

[2] Cho, S. and Balke, W.T. Order-preserving optimization of twig queries with structural preferences. In *Proc. of the Int. Database Engineering & Applications Symposium*, Coimbra, Portugal, 2008.

[3] Chomicki, J. Preference formulas in relational queries. In *ACM Transactions on Database Systems*, Vol. 28(4), ACM, 2003.

[4] Kießling, W. Foundations of preferences in database systems. In *Proc. of the Int. Conf. on Very Large Databases*, Hong Kong, China, 2002.