Rank-aware XML Data Model and Algebra: Towards Unifying Exact Match and Similar Match in XML

WANG NING XU DE
Computer Science Department
Beijing Jiaotong University
School of Computer Science & Information Technology, Beijing Jiaotong University, Beijing,100044
CHINA

Abstract: In this paper, we present rank-aware XML data model and algebra, which aims at providing a seamless support and integration of ranked queries with precise queries. Our data model is based on ranked XML trees by extending the semantics of XML trees to be rank-aware. A collection of ranked XML trees which can be ordered by their ranks is modeled as a sequence, which is the basis for ranked XML operations. With ranked XML tree model, we extend XML algebra by introducing new ranking operator and extending ordinary XML operators to fully support top-k queries. A set of algebraic laws is also defined in this paper for rewriting and optimizing top-k queries.

Key-Words: XML, data model, algebra, Top-k query processing, rank, query optimization

1 Introduction

Until now, the traditional Database management system(DB), Multimedia data management system (MDB) and Information Retrieval system(IR) have evolved largely independently of each other. The traditional DB, which has mostly focused on managing highly structured data, has developed sophisticated techniques for efficiently processing complex and precise queries over this data. In contrast, the IR and MDB, which have focused on searching semi-structured and even unstructured data, have developed various techniques for ranking query results and evaluating their effectiveness. However, there has been no single unified system model for managing structured, semi-structured and unstructured data, and processing both precise and ranked queries [1].

In fact, recent trends in research demonstrate a growing interest in adopting MDB and IR techniques in DBs and vice versa [2]. Efficient evaluations of ranked queries in relational database systems have recently gained the attention of the research community. Most of the available solutions to supporting ranked queries are in the middleware scenario [3], or in RDBMS only focusing on specific types of operators and queries outside the core of query engines[4], while a few of them support top-k queries in the relational query engine[5][6].

However, relational DB has advantages mainly for managing structured data, while multimedia data and full-text data can only be stored as BLOB, CLOB. In fact, with the emergency of MPEG-7, abundant of multimedia data in XML format attracts researchers' attention. One of the key benefits of XML is its ability to represent a mix of structured and unstructured data. TeXQuery[7] is the precursor of the full-text language extensions to XPath2.0 and XQuery1.0 currently being developed by the W3C, which provides a rich set of full-text search primitives and scoring construct without extension to XQuery data model. The solution of TeXQuery is specific to text search and lack of fundamental support of ranked queries, while some solutions focus on developing algorithms for efficiently computing top-k matches in XML [8].

Fundamental support of ranked queries is lacking mainly because neither XML data model nor XML algebra has the notion for ranking. Therefore, supporting ranked queries in XML DBMS as a first-class query type inside the core of query engines is a significant research challenge. In this paper, we present rank-aware XML data model and algebra, which aims at providing a seamless support and integration of ranked queries with precise queries.

The rest of the paper is organized as follows. We start in section 2 by giving examples to illustrate our motivation. Section 3 introduces rank-aware XML data model. We presents rank-aware XML algebra and algebraic laws as the base for query optimization in section 4.Finally, we conclude the paper in section 6.

2 Motivation

Ranked queries often request the top k results. Suppose we have a XML database for house (Fig.1 shows two houses of them), following is an example of top-k query.

Example 2.1 Consider user Linda, who wants to rent a house in Boston. She needs a house with size greater than 200m^2 and price less than 400$. Further, to rank the qualified results, she specifies several ranking criteria: near the park A and with the appearance as the photograph B that she likes.

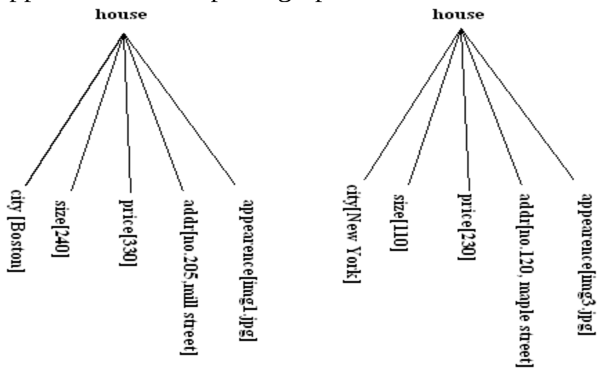

Fig.1 a XML database for house

To formulate the query in this example, we have three filtering predicates (FP) and two ranking predicates (RP) as follows:

FP1: /house/city="Boston"
FP2: /house/size>200
FP3: /house/price<400
RP1: near (/house/addr, A)
RP2: similar (/house/appearance, B)

Filtering predicates which return Boolean values will be used as filter conditions to get the exact match results, while ranking predicates which return numeric scores can be used to rank the results.The overall scoring function can be specified by Linda as summing up the scores of RP1 and RP2.

In order to get the house according to Linda's interests, the query processor can first evaluate the filter conditions and then sort the results according to scoring function just as the processing method in most relational databases. However, processing in this way suffers from the following two problems.

First, because users may usually be interested in the top-k results, providing all the possible results is unnecessary and time consuming.

Second, sorting the results at final stage means that query processors must evaluate all ranking predicates for every possible result, which is usually very expensive.

From the analysis above, we can conclude that the evaluation cost of the ranked queries will be greatly reduced if the irrelevant results can be pruned as early as possible. The general approach we put forward for supporting ranking in XML query engines is based on extending XML model and XML algebra to be rank-aware. We will introduce a new ranking operator and extend ordinary XML operators to fully support ranked queries, while a set of algebraic laws is also defined so that ranking operator can be interleaved with other operators instead of always being processed after filtering.

3 Rank-aware XML Data Model

In this section, we define ceiling score and rank pattern first, and then extend the XML data model with rank.

3.1 Ceiling Score

We have mentioned ranking predicates, which are functions to return numeric scores. Based on ranking predicates, a rank expression can be defined.

Definition 3.1.1 (rank expression) Given a set of ranking predicates $RP=\{rp_1,rp_2,\ldots,rp_n\}$, a rank expression F_{RP} is a monotonic scoring function that maps a set of numeric scores for member of RP to a overall numeric score.

Definition 3.1.2 (score of XML data tree) Given a XML data tree T and a rank expression F_{RP}, $RP=\{rp_1,rp_2,\ldots,rp_n\}$, we define the score of F_{RP} under T, denoted $S(F_{RP},T)$, as

$$S(F_{RP}, T) = F_{RP}(rp_1[T], rp_2[T],\ldots,rp_n[T])$$

In above definition, $rp_i[T](i\in[1,n])$ represents the score of ranking predicate rp_i evaluated on T. For simplicity, we assume the $rp_i[T]<=1$ in this paper.

The score of a rank expression F_{RP} can be calculated precisely only if every ranking predicate in RP be evaluated. However, in order to optimize query with rank expression efficiently, ranking predicates need to be evaluated separately and interleaved with other operators. At some intermediate stage when we do not have the complete scores of all the ranking predicates, we also want to define a partial ranking of results by their current incomplete scores, so that the resulted order is consistent with the desired order of further processing. Ceiling score is defined to order the output intermediate results to subsequent operations.

Definition 3.1.3 (ceiling score) With respect to a XML data tree T, a rank expression F_{RP} , and a set of evaluated ranking predicates $EP\subseteq RP$, we can define the ceiling score $\hat{S}(F_{RP},T)$ of T under F_{RP} as

$$\hat{S}(F_{RP}, T)=F_{RP}(rp_1[T],rp_2[T],\ldots,rp_n[T])$$
$$\forall i \in [1,n], rp_i[T]=1 \text{ if } rp_i \notin EP$$

Because F_{RP} is monotonic, ceiling score is the max possible score of a XML data tree which can archive.With ceiling scores, XML data trees can be ordered.

Theorem 3.1 Given a rank expression F_{RP} and two XML data trees T_1, T_2, if $\hat{S}$ (F_{RP},T_1) > $\hat{S}$ (F_{RP},T_2), then T_1 must be processed before T_2 when we need further process for answering the query.

The proof is intuitive, so we will omit it for saving space.

3.2 Ranked XML Tree

To fundamentally support ranking, we need extend XML data model to make it rank-aware. In order to evaluate the ranking predicates in a rank expression incrementally step by step, we use ceiling scores to order the intermediate results for further processing. Since tree model is natural to describe XML document, XML algebras such as TAX[9] based itself on a collection of XML data trees. XML data trees in a collection are unordered, which is suitable for precise queries. For top-k queries, data trees should be ordered according to its ceiling scores for pruning irrelevant answers as early as possible during the evaluation process. From now on, we will add rank notion to XML data tree.

Definition 3.2.1 (rank pattern) With respect to a rank expression F_{RP}, RP={rp_1,rp_2,…,rp_n}, we can define a rank pattern RPT(F_{RP}) as a pair (T, R), where T=(V,E) is a node-labeled and edge-labeled tree such that:

(1) Each node in V which represents nodes of interest for ranking has a distinct integer as its label;

(2) Each edge is either labeled pc (for parent-child) or ad (for ancestor-descendant)

(3) R is a boolean combination of predicates applicable to nodes according to ranking predicates in RP and rank expression F_{RP}.

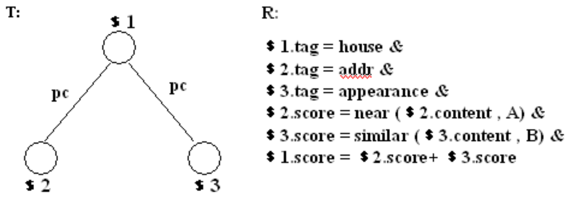
T:
$ 1
pc
pc
$ 2
$ 3
R:
$ 1.tag = house &
$ 2.tag = addr &
$ 3.tag = appearance &
$ 2.score = near ($ 2.content , A) &
$ 3.score = similar ($ 3.content , B) &
$ 1.score = $ 2.score+ $ 3.score

Fig.2 an example for rank pattern

The rank pattern of query in example 2.1 can be represented as Fig.2. Compared to pattern in TAX, a rank pattern here can not only provide nodes of interest for similar match by T but also provide the algorithm for calculating scores of those nodes by R.

Definition 3.2.2 (rank tree) With a XML data tree T, a rank pattern RPT(F_{RP}), and a set of evaluated ranking predicates P$\subseteq$RP, we can define a rank tree RT of T under RPT and P as *RT*(RPT,P)[T] such that:

(1) RT preserves the structure of RPT;

(2) Every leaf node in RT keeps the score of corresponding ranking predicates in P which have been evaluated on T, while the root node in RT keeps the ceiling score of T.

A rank tree always goes with a data tree, and it keeps the trail of evaluation for ranking predicates. For a ranking predicate which has not been evaluated, the corresponding node in rank tree keeps 1 as its score.

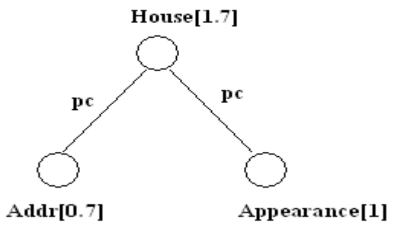

Fig.3 an example for rank tree

We have given example 2.1 which has two ranking predicates RP1 and RP2. Fig.3 shows a rank tree according to the first data tree in XML database as Fig.1 and the rank pattern in Fig.2 when only RP1 has been evaluated. From the figure, we can know that the score of RP1 is 0.7 and the score of unevaluated ranking predicate RP2 is 1, so the ceiling score of the corresponding data tree is 1.7.

Definition 3.2.3 (Ranked XML tree) Given a XML data tree T, a rank pattern RPT(F_{RP}), and a set of evaluated ranking predicates P$\subseteq$RP, we can define a ranked XML tree RXT(RPT,P)[T] as a pair (T,RT), where RT is a rank tree of T under RPT and P. We call T the data tree of RXT which is denoted as DT(RXT).The rank of RXT is defined as the ceiling score of T which is denoted as *R*(RPT,P)[RXT].

In order to unify exact match and similar match in XML, we treat a ranked XML tree as a fundamental unit, similar to a tuple in RDBMS.

4 Rank-aware XML Algebra

To enable rank-based XML query processing and optimization, we propose rank-aware XML algebra (RXA) in which every operation is based on a sequence of ranked XML trees. Compared to other XML tree algebra, such as TAX which takes a collection of unordered data trees as input, RXA uses a sequence of ranked XML trees as operand. We use “sequence” to emphasize the order in ranked XML trees. As we have mentioned, ranked XML trees are ordered according to their ceiling scores.

4.1 Algebraic Operation

Before we could introduce a new operator for ranking and extend ordinary XML algebra operators with ranking concept, a relationship named rank order relationship will be given first.

4.1.1 Rank Order Relationship

Definition 4.1.1 (Rank Order Relationship) A rank order relationship $\prec_R$ is defined over a sequence of ranked XML trees S with a rank pattern RPT(F_{RP}) and a set of evaluated ranking predicates P$\subseteq$RP such that:

$\forall$ RXT_1, $RXT_2$$\in$S, $RXT_1 \prec_R RXT_2$
iff *R*(RPT, P) [RXT_1] $<$ *R*(RPT, P) [RXT_2]

Ceiling scores are the basis for ordering ranked XML trees. Because a ranked XML tree with higher rank will always be processed before the others with lower ranks, it is possible that we can get top-k results without processing every ranked tree.

4.1.2 Ranking Operator

Obviously, ranking is a necessary and important operation for top-k query. We define a new ranking operator *r* which is able to evaluate the ranking predicates in a rank expression one at a time.

With a sequence of ranked XML trees S according to a rank pattern RPT(F_{RP}), a set of ranking predicates P evaluated , and a set of ranking predicates {p_1, p_2,…, p_j}$\subseteq$RP, ranking operation *r*[p_1, p_2,…, p_j](S) can output a sequence of ranked XML trees such that:

(1) RXT $\in$*r*[p_1, p_2,…, p_j](S), RXT=(T,RT) iff
$\exists$ $RXT_1$$\in$S, T =DT($RXT_1$) $\wedge$ RT=*RT* (RPT, P$\cup${p_1, p_2,…, p_j})[T]

(2) $\forall$ RXT_1,$RXT_2$$\in$*r*[$p_1$, p_2,…, p_j](S),
$RXT_1 \prec_R RXT_2$ iff
R(RPT,P$\cup${p_1,p_2,…,p_j})[RXT_1]<
R(RPT,P$\cup${p_1,p_2,…, p_j})[RXT_2]

Through ranking operation, a sequence of ranked XML trees can be reordered by their ranks.

4.1.3 Extended Operators

Ranking operator is a critical basis of RXA, however, we also should extend the traditional XML algebra operators to be rank-aware. For the limitation of space, we only select some typical operators which are common in XML algebra to explain our extensions. Our extensions in the following paper is based on TAX and OrientXA[10], where pattern is an important concept for extracting nodes of interest. For simplicity, we reserve the key concepts but leave some parameters such as adornment SL, projection List PL out in the following.

1. Rank-aware Selection Operator

Given a sequence of ranked XML trees S, a rank-aware selection operation based on a rank pattern RPT and a pattern PAT can be represented as

σ_R[PAT, RPT](S)

The result of above operation is a sequence of ranked XML trees, in which every data tree is a witness tree under pattern PAT and its rank tree is selected accordingly from input sequence S.

2. Rank-aware Projection Operator

Given a sequence of ranked XML trees S, a rank-aware projection operation based on a rank pattern RPT and a pattern PAT can be represented as

π_R[PAT, RPT](S)

The result of above operation is also a sequence of ranked XML trees, in which every data tree is an element of TAX projection result under pattern PAT based on the set of data trees in S and its rank tree is selected accordingly from input sequence S.

In fact, both rank-aware selection and projection operation keep the order of output sequence consistent with the input because no other ranking predicates are evaluated during the operations.

3. Rank-aware Join Operator

Given two sequences of ranked XML trees S_1 and S_2, two rank patterns RPT_1 and RPT_2 corresponding to S_1 and S_2 respectively, a join predicate C, a rank-aware join operation can be represented as

$\bowtie_R$[C, RPT_1, RPT_2](S_1,S_2)

The result of above operation is a sequence of ranked XML trees. Every data tree T in the result is an element of TAX join result under join predicate C based on the sets of data trees in S_1 and S_2. Every rank tree RT with T can be constructed according to the rank trees in S_1 and S_2 as follows:

(1) If $\exists$ $RXT_1$$\in$$S_1$, $\exists$ $RXT_2$$\in$$S_2$, RXT_1=(T_1,RT_1) , RXT_2=(T_2,RT_2), and T is the join result from T_1 and T_2, then RT has a new root with RT_1 as its left child and RT_2 as its right child.

(2) The ceiling score of RT kept in its root will be recalculated according to the union of evaluated ranking predicates set from S_1 and S_2

4. Rank-aware Construction Operator

Construction operator is necessary for user to construct query results in XQuery. According to the OrientXA, construction operation has two patterns, input pattern and output pattern, as its parameters. To make construction operation rank-aware, we add two parameters as well, which are input rank pattern and output rank pattern.

Given a sequence of ranked XML trees S, an input pattern PAT_I ,an output pattern PAT_O, an input rank pattern RPT_I, and an output rank pattern RPT_O, a rank-aware construction operation can be represented as

χ_R[PAT_I, PAT_O,RPT_I, RPT_O](S)

The result of above operation is also a sequence of ranked XML trees. Every data tree T in the result is constructed according to the output pattern PAT_O

based on the sets of data trees in S. Every rank tree RT with T can be constructed according to the output rank pattern RPT_O.

4.2 Algebraic Laws

The aim of rank-aware XML data model and algebra is to support ranked queries in XML DBMS as a first-class query type inside the core of query engines. Query optimizers essentially rely on algebraic equivalences to enumerate or transform query plans in search of efficient ones. In this subsection, we will give algebraic equivalences based on RXA.

In order to reduce the cost of ranked queries, we should evaluate the ranking predicates incrementally instead of evaluating all of them and ordering the results at final stage. Query optimizers need algebraic equivalences to separate ranking predicates with each other and interleave ranking operator with other operators. To save space, we only concentrate on some of the equivalences relevant to ranking.

1. Separating law for ranking operator

(1) $r\,[p_1, p_2, \ldots p_j](S) \equiv r[p_1](r[p_2](\ldots r\,[p_j](S))\ldots))$

2. Commutative laws

(1) $r\,[p_1](\,r\,[p_2](\,S)) \equiv r\,[p_2](\,r\,[p_1](\,S))$

(2) $\sigma_R[PAT, RPT](\,r\,[p_1](\,S))$
$\equiv r\,[p_1](\sigma_R[PAT, RPT]\,(\,S))$

(3) $\pi_R[PAT, RPT]\,(\,r\,[p_1](\,S))$
$\equiv r\,[p_1]\,(\,\pi_R[PAT, RPT]\,(S))$

(4) $\chi_R[PAT_I, PAT_O, RPT_I, RPT_O](r[p_1]\,(\,S))$
$\equiv r[p_1](\,\chi_R[PAT_I, PAT_O, RPT_I, RPT_O]\,(\,S))$

3. Pushing ranking operator over join operator

(1) $r\,[p_1](\triangleright\triangleleft_R[C, RPT_1, RPT_2](\,S_1, S_2))$
$\equiv \triangleright\triangleleft_R[C, RPT_1, RPT_2](\,r\,[p_1]\,(\,S_1)),$
if only data tree of S_1 has attributes in p_1

(2) $r\,[p_1](\triangleright\triangleleft_R[C, RPT_1, RPT_2](\,S_1, S_2)) \equiv$
$\triangleright\triangleleft_R[C, RPT_1, RPT_2](r[p_1](S_1), r[p_1](S_2)),$
if data trees of both S_1 and S_2 have attributes in p_1

Above laws allow us to separate the ranking operation with several predicates into a series of ranking operations. After that, ranking operations can swap with other operations. To be rank-aware, those algebraic equivalences reserve not only the same membership in sequence but also the same order in sequence.

5 Conclusion

We introduce our solution for unifying exact match and similar match in XML. As the foundation of our work, we first extend XML data model to make it rank aware. The extended model is based on ranked XML trees whose ranks are defined as max possible scores for a set of ranking predicates under the circumstance that some of predicates have been evaluated. A collection of ranked XML trees which can be ordered by their ranks has been modeled as a sequence, which is the basis for XML operations. Second, we extend XML algebra which captures the ranking property with ranked XML tree model and introduce new and extended operators to fully support top-k queries. Third, we also define a set of algebraic laws that would be used for rewriting and optimizing top-k queries.

While we believe that the definition of rank-aware XML data model and algebra is a significant step towards unifying precise queries and ranked queries, we are currently working on defining physical algebra which can be mapped from logical algebra in order to realize query optimization. Work on developing proper data structures and algorithms for optimization is underway.

References:

[1] Sihem Amer-Yahia, Pat Case, Report on the DB/IR Panel at SIGMOD 2005, SIGMOD Record, Vol.34, No.4, 2005,pp.71-74.

[2] Sihem Amer-Yahia, Jayavel Shanmugasundaram, XML Full-Text Search: Challenges and Opportunities, VLDB, 2005, pp.1368-1368.

[3] K.C.-C. Chang and S. Hwang. Minimal probing: Supporting Extensive Predicates for Top-k Queries, SIGMOD, 2002, pp.346-357.

[4] S. Guha, N. Koudas, A. Marathe, and D. Srivastava. Merging the Results of Approximate Match Operations, VLDB, 2004, pp.636-647.

[5] Chengkai Li, Kevin Chen-chuan Chang, Ihab F. IIyas, Sumin Song, RankSQL: Query Algebra and Optimization for Relational Top-k Queries, SIGMOD, 2005, pp.131-142.

[6] Zhen Zhang, Seung-won Hwang, Kevin Chen-Chuan Chang, Boolean+Ranking: Querying a Database by K-Constrained Optimization, SIGMOD, 2006, pp.359-370.

[7] S. Amer-Yahia, C. Botev, J. Shanmugasundaram, TeXQuery: A Full-Text Search Extension to XQuery. WWW2004, New York, USA, 2004, pp.583-594.

[8] A. Marian, S. Amer-Yahia , N. Koudas, D. Srivastava, Adaptive Processing of Top-k Queries in XML, ICDE, 2005, pp. 162-173.

[9] Jagadish VH, Al-Khalifa S, Lakshmanan L, Srivastava D, Thompson K, TAX: A Tree Algebra for XML, DBLP, 2001, pp.149-164.

[10] Meng XF, Luo DF, Jiang Y, Wang Y, OrientXA: A Effective XQuery Algebra, Journal of Software, Vol.15, No.11, 2004, pp.1648-1660.