PowerDB-XML: a Platform for Data-Centric and Document-Centric XML Processing

Torsten Grabs and Hans-Jörg Schek

Database Research Group,
Institute of Information Systems,
Swiss Federal Institute of Technology Zurich, Switzerland,
{grabs,schek}@inf.ethz.ch

Abstract. Relational database systems are well-suited as a platform for data-centric XML processing. Data-centric applications process regularly structured XML documents using precise predicates. However, these approaches come too short when XML applications also require document-centric processing, i.e., processing of less rigidly structured documents using vague predicates in the sense of information retrieval. The PowerDB-XML project at ETH Zurich aims to address this drawback and to cover both these types of XML applications on a single platform. In this paper, we investigate the requirements of document-centric XML processing and propose to refine state-of-the-art retrieval models for unstructured flat document such that they meet the flexibility of the XML format. To do so, we rely on so-called *query-specific statistics* computed dynamically at query runtime to reflect the query scope. Moreover, we show that document-centric XML processing is efficiently feasible using relational database systems for storage management and standard SQL. This allows us to combine document-centric processing with data-centric XML-to-database mappings. Our XML engine named PowerDB-XML therefore supports the full range of XML applications on the same integrated platform.

1 Introduction

The eXtended Markup Language XML [30] is highly successful as a format for data interchange and data representation. The reason for this is the high flexibility of its underlying semistructured data model [1]. The success of XML is reflected by the interest it has received by database research following its recommendation by the W3C in 1998. However, this previous stream of research has mainly focused on *data-centric XML processing*, i.e., processing of XML documents with well-defined regular structure and queries with precise predicates. This has led to important results which, for instance, allow to map data-centric XML processing onto relational database systems. XML and its flexible data model however cover a much broader range of applications, namely the full range from data-centric to document-centric processing. *Document-centric XML processing* deals with less rigidly structured documents. In contrast to data-centric

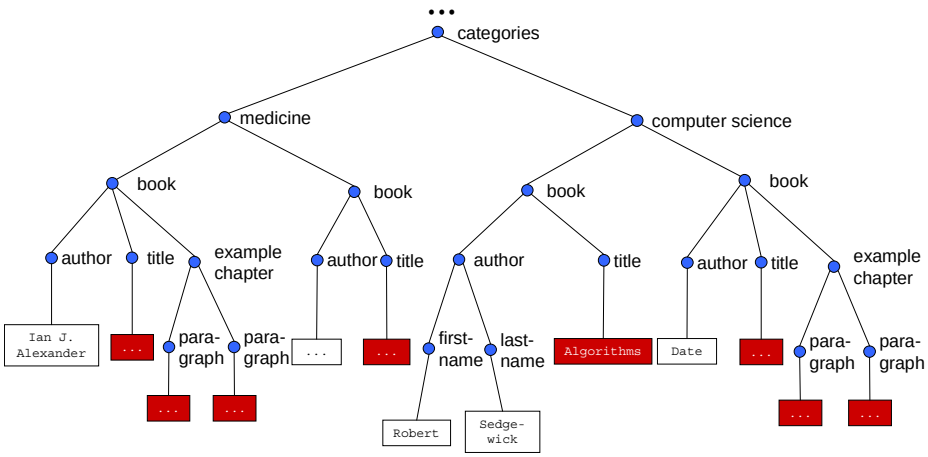

Fig. 1. Exemplary XML document with textual content represented as shaded boxes

processing, document-centric applications require processing of vague predicates, i.e., queries expressing information needs in the sense of information retrieval (IR for short).

Data-centric approaches proposed in previous work on XML processing do not cover document-centric requirements. In addition, conventional ranked retrieval on text documents is not directly applicable to XML documents because of the flexibility of the XML data model: users want to exploit this flexibility when posing their queries. Such queries usually rely on path expressions to dynamically combine one or several element types to the scope of a query. This is in contrast to conventional IR where the retrieval granularity is restricted either to complete documents or to predefined fields such as abstract or title. Dynamically defined query scopes with XML retrieval however affect retrieval, their ranking techniques, and in particular local and global IR statistics. *Local IR statistics* represent the importance of a word or term in a given text. *Term frequencies*, i.e., the number of occurences of a certain term in a given document are local statistics with the vector space retrieval model, for instance. *Global IR statistics* in turn reflect the importance of a word or a term with respect to the document collection as a whole. Taking again vector space retrieval as an example, *document frequencies*, i.e., the number of documents a given term appears in, are global statistics. State-of-the-art retrieval models such as vector space retrieval combine both local and global statistics to compute the ranked result of an IR query.

The following discussion takes vector space retrieval as a running example and illustrates shortcomings of conventional IR statistics in the context of XML retrieval.

Consider the example XML document shown in Fig. 1. The document contains information about books from the domains of medicine and computer science. Such a document often is the only document in the collection, i.e., all infor-

mation is stored in a single XML document. This represents a typical situation with many practical settings. Consequently, conventional vector space document frequencies equal either 1 or 0: 1 for all terms occurring in the document and 0 otherwise. Hence, the intention of global statistics to discriminate important and less important terms is lost when using conventional IR statistics for XML retrieval. A further observation is that conventional term frequencies do not reflect different query scopes: term frequencies are computed for the document as a whole. XML retrieval however often restricts search to certain sub-trees in the XML document, e.g., the computer science or medicine branch of the example document shown in Fig. 1. Our bottomline therefore is that conventional IR statistics need to be refined for flexible retrieval from XML documents.

The objective of our work is twofold: we want to address the aforementioned issues regarding IR statistics with so-called *query-specific IR statistics*. They are computed on-the-fly, i.e., at query processing time, and both local and global statistics reflect the scope of the query. Our second objective is to make respective document-centric XML retrieval functionality available on a platform such as a relational database system that is also well-suited for data-centric XML processing. Regarding these objectives, this current paper makes the following contributions: based on previous work [16, 17], we define query-specific IR statistics for flexible XML retrieval. Moreover, we show how to realize document-centric XML processing with query-specific statistics on top of relational database systems in combination with data-centric XML processing. Our overall contribution is our XML engine called PowerDB-XML. Based on relational database systems for storage management, it realizes the envisioned platform for joint data-centric and document-centric XML processing.

The remainder of the paper discusses PowerDB-XML's approach to joint data-centric and document-centric XML processing on top of relational database systems. The following section (Sect. 2) covers related work. In Sect. 3, we discuss flexible retrieval on XML documents. The section defines query-specific statistics and explains them in more detail using vector space retrieval and *tf idf* ranking as an example. Section 4 in turn describes the system architecture of PowerDB-XML. Using again vector space retrieval, it explains how to extend relational data-centric XML mappings in order to store index data for efficient flexible retrieval from XML documents using query-specific statistics. Section 5 then discusses processing of document-centric requests over the XML collection stored. Special interest is paid to computing query-specific IR statistics dynamically, i.e., at query runtime, from the underlying IR index data using standard SQL. Section 6 concludes the paper.

2 Related Work

Data-centric XML processing has received much interest by database research after XML has been recommended by the W3C in 1998. This has led to important results such as query languages for data-centric XML processing (XPath [28] and XQuery [29] among others). Besides query languages, several data-centric

mappings for XML documents to relational databases have been proposed, e.g., EDGE and BINARY [7], STORED [5], or LegoDB [3, 2]. In addition, several approaches have been devised to map XML query languages to relational storage and SQL [4, 6, 21, 26, 25]. Relational database systems are therefore well-suited as a platform for data-centric XML processing.

Recent work has extended this previous stream of research to keyword search on XML documents. Building on relational database technology, it has proposed efficient implementations of inverted list storage and query processing [8, 19]. While already refining the retrieval granularity to XML elements, this previous work has still focused on simple Boolean retrieval models. Information retrieval researchers instead have investigated document-centric XML processing using state-of-the-art retrieval models such as vector space or probabilistic retrieval models. An important observation there was that conventional retrieval techniques are not directly applicable to XML retrieval [13, 11]. This in particular affects IR statistics used in ranked and weighted retrieval which heavily rely on the retrieval granularities supported. To increase the flexibility of retrieval granularities for searching XML, Fuhr et al. group XML elements (at the instance level) to so-called *indexing nodes* [13]. They constitute the granularity of ranking with their approach while IR statistics such as *idf* term weights are derived for the collection as a whole. The drawback of the approach is that the assignment of XML elements to indexing nodes is static. Users cannot retrieve dynamically, i.e., at query time, from arbitrary combinations of element types. Moreover, this can lead to inconsistent rankings when users restrict the scopes of their queries to element types that do not directly correspond to indexing nodes and whose IR statistics and especially term distributions differ from the collection-wide ones. Our previous work [16] has already investigated similar issues in the context of flat document text retrieval from different domains: queries may cover one or several domains in a single query and ranking for such queries depends on the query scope. Based on this previous work, [17] proposes XML elements as the granularity of retrieval results and refines IR statistics for *tf idf* ranking [22] in this respect. Our approach derives the IR statistics appropriate to the scope of the queries in the XML documents dynamically at query runtime. This current paper extends this previous work by an efficient relational implementation which allows to combine both data-centric and document-centric XML processing on relational database systems.

3 Flexible XML Retrieval with Query-Specific Statistics

Conventional IR statistics for ranked and weighted retrieval come too short for XML retrieval with flexible retrieval granularities [13]. This section extends conventional textual information retrieval models on flat documents to flexible retrieval on semistructured XML documents. A focus of our discussion is on vector space retrieval and to refine it with query-specific statistics. Retrieval with query-specific statistics also serves as the basic component for document-centric processing with PowerDB-XML.

3.1 Retrieval with the Conventional Vector Space Model

Conventional vector space retrieval assumes flat document texts, i.e., documents and queries are unstructured text. Like many other retrieval techniques, vector space retrieval represents text as a 'bag of words'. The words contained in the text are obtained by IR functionality such as term extraction, stopword elimination, and stemming. The intuition of vector space retrieval is to map both document and query texts to n-dimensional vectors d and q, respectively. n stands for the number of distinct terms, i.e., the size of the vocabulary of the document collection. A text is mapped to such a vector as follows: each position i $(0 < i \leq n)$ of v represents the i-th term of the vocabulary and stores the *term frequency*, i.e., the number of occurrences of t in the text. A query text is mapped analogously to q. Given some query vector q and a set of document vectors C, the document with $d \in C$ that has the smallest distance to or smallest angle with q is deemed most relevant to the query. More precisely, computation of relevance or retrieval status value (rsv) is a function of the vectors q and d in the n-dimensional space. Different functions are conceivable such as the inner product of vectors or the cosine measure. The remainder of this paper builds on the popular so-called *tf idf ranking function* [24]. *tf idf* ranking constitutes a special case of vector space retrieval. Compared to other ranking measures used with vector space retrieval, it has the advantage to approximate the importance of terms regarding a document collection. This importance is represented by the so-called *inverted document frequency* of terms, or *idf* for short. The *idf* of a term t is defined as $idf(t) = \log \frac{N}{df(t)}$, where N stands for the number of documents in the collection and $df(t)$ is the number of documents that contain the term (the so-called *document frequency* of t). Given a document vector d and a query vector q, the retrieval status value $rsv(d, q)$ is defined as follows:

$$rsv(d, q) = \sum_{t \in terms(q)} tf(t, d)\ idf(t)^2\ tf(t, q) \tag{1}$$

Going over the document collection C and computing $rsv(d, q)$ for each document-query-pair with $d \in C$ yields the ranking, i.e., the result for the query.

In contrast to Boolean retrieval, ranked retrieval models and in particular vector space retrieval assume that documents are flat, i.e., unstructured information. Therefore, a straight-forward extension to cover retrieval from semistructured data such as XML documents and to refer to the document structure is not obvious. But, ranked retrieval models are known to yield superior retrieval results [9, 23]. The following paragraphs investigate this problem in more detail and present an approach that combines flexible retrieval with result ranking from vector space retrieval.

3.2 Document and Query Model for Flexible XML Retrieval

In the context of this paper, XML documents are represented as trees. We rely on the tree structures defined by the W3C XPath Recommendation [28]. This

yields tree representations of XML documents such as the one shown in Fig. 1. Obviously, all textual content of a document is located in the leaf nodes of the tree (shaded boxes in the figure). For ease of presentation, we further assume that the collection comprises only a single XML document – a situation one frequently encounters also in practical settings. Note that this is not a restriction: it is always possible to add a virtual root node to compose several XML documents into a single tree representation such that the subtrees of the virtual root are the original XML documents. Moreover, we define the *collection structure* as a complete and concise summary of the structure of the XML documents in the document collection such as the DataGuide [15].

Flexible retrieval on XML now aims to identify those subtrees in the XML document that cover the user's information need. The granularity of retrieval in our model are the nodes of the tree representation, i.e., subtrees of the XML document. The result of a query is a ranked list of such subtrees. Users define their queries using so-called *structure constraints* and *content constraints*.

Structure constraints define the scope, i.e., the granularity, of the query. With our query model, the granularity of a query is defined by a label path. Taking the XML document in Fig. 1 for instance, the path /bookstore/medicine/book defines a query scope. The extension of the query scope comprises all nodes in the XML document tree that have the same path originating at the root node. The extension of /bookstore/medicine/book comprises two instances – the first and the second medicine book in the document. Users formulate their structure constraints using path expressions. With the XPath syntax [28] and the XML document in Fig. 1, the XPath expression //book for instance yields a query granularity comprising /bookstore/medicine/book and /bookstore/computer-science/book.

Content constraints in turn work on the actual XML elements in the query scope. We distinguish between so-called *vague content constraints* and *precise content constraints*. A vague content constraint defines a ranking over the XML element instances in the query scope. A precise content constraint in turn defines an additional selection predicate over the result of the ranking. In the following, we exclude precise content constraints from our discussion and focus instead on vague content constraints for ranked text search.

3.3 Result Ranking with Query-Specific Statistics

In our previous discussion, we have refined the retrieval granularity of XML retrieval to XML elements. Hence, our query model returns XML elements e in the query result, and we have to adapt the ranking function accordingly:

$$rsv(e,q) = \sum_{t \in terms(q)} tf(t,e)\ ief(t)^2\ tf(t,q) \tag{2}$$

In contrast to Equation 1, the ranking function now computes a ranking over XML elements e under a query text q. Moreover, term frequencies tf and inverted element frequencies ief now work at the granularity of XML elements. The following paragraphs investigate the effects of these adaptations in more detail and refine global and local IR statistics to query-specific statistics.

Global IR Statistics. Different parts of a single XML document may have content from different domains. Figure 1 illustrates this with the different branches of the bookstore – one for medicine books and one for computer science books. Intuitively, the term 'computer' is more significant for books in the medicine branch than in the computer science branch. IR statistics should reflect this when users query different branches of the collection structure. The first – and simplest case – is when a query goes to a single branch of the collection structure. We denote this as *single-category retrieval*. In this case, the query-specific global statistics are simply computed from the textual content of the collection structure branch where the query goes to. The following example illustrates this retrieval type.

Example 1 (Single-Category Retrieval). Consider a user searching for relevant books in the computer science branch of the example document in Fig. 1. Obviously, he restricts his queries to books from this particular category. Thus, it is not appropriate to process this query with term weights derived from both the categories medicine and computer science in combination. This is because the document frequencies in medicine may skew the overall term weights such that a ranking with term weights for computer science in isolation increases retrieval quality.

Taking again our running example of vector space retrieval with $tf\,idf$ ranking, global IR statistics are the (inverted) element frequencies with respect to the single branch of the collection structure covered by the query. We therefore define the element frequency $ef_{cat}(t)$ of a term t with respect to a branch cat of the collection structure as the number of XML element sub-trees that t occurs in. More formally:

$$ef_{cat}(t) = \sum_{e \in cat} \chi(t, e) \tag{3}$$

with $\chi(t, e)$ defined as follows:

$$\chi(t, e) = \begin{cases} 1, \text{ if } \sum_{se \in SE(e)} tf(t, se) > 0 \\ \\ 0, \text{ otherwise} \end{cases} \tag{4}$$

Thus, $\chi(t, e)$ is 1 if at least e or one of its sub-elements se contains t.

Now think of another user who wants to process a query on several categories, i.e., on several non-overlapping branches of the collection structure. We call such queries *multi-category retrieval*. In other words, a multi-category query goes over one or several single-category query scopes. The difficulty with this type of queries is again that conventional IR statistics are not meaningful in the XML context, as already argued above. A more promising alternative in turn is to rely on query-specific statistics reflecting the scope of the query. The following example illustrates multi-category retrieval.

Example 2 (Multi-Category Retrieval). Recall the XML document from the previous example (cf. Fig. 1). The document in the figure reflects the different categories of books such as medicine or computer science with separate element

types for the respective categories. Think of a user who does not care to which category a book belongs, as long as it covers the information need expressed in his query. The granularity of his query are all categories. Hence, the query is an example of *multi-category retrieval* which requires query-specific statistics. Taking again the document in Fig. 1, this means statistics must be derived from both categories medicine and computer science in combination.

With vector space retrieval and *tf idf* ranking, we define the global query-specific IR statistics as follows given a query scope *mcat*: the multi-category element frequency $ef_{mcat}(t)$ of a term t is the number of sub-trees in the XML documents t occurs in. Given this definition, the following equation holds between a multi-category query scope M_q and the single-categories it comprises.

$$ef_{mcat}(t, \mathcal{M}_q) = \sum_{cat \in \mathcal{M}_q} ef_{cat}(t) \tag{5}$$

This yields the multi-category inverted document frequency:

$$ief_{mcat}(t, \mathcal{M}_q) = \log \frac{\sum_{cat \in \mathcal{M}_q} N_{cat}}{\sum_{cat \in \mathcal{M}_q} ef_{cat}(t)} \tag{6}$$

Local IR Statistics. XML allows to hierarchically structure information within a document such that each document has a tree structure. Users want to refer to this structure when searching for relevant information. The intuition behind this is that an XML element is composed from different parts, i.e., its child elements. For instance, a chapter element may comprise a title and one or several paragraph elements. This is an issue since the children elements may contribute to the content of an XML element by different degrees. Fuhr at al. for instance reflect the importance of such composition relationships with so-called augmentation weights that downweigh statistics when propagating terms along composition relationships [13]. This also affects relevance-ranking for XML retrieval, as the following example shows.

Example 3 (Nested Retrieval). Consider again the XML document shown in Fig. 1. Think of a query searching for relevant book elements in the medicine branch. Such a query has to process content that is hierarchically structured: the title elements as well as the paragraph elements describe a particular book element. Intuitively, content that occurs in the title element is deemed more important than that in the paragraphs of the example chapter, and relevance ranking for books should reflect this.

Hierarchical document structure in combination with augmentation affects local IR statistics only. Consequently, term frequencies are augmented, i.e., downweighed by a factor $aw \in [0; 1]$ when propagating them upwards from a sub-element se to an ancestor element e in the document hierarchy. This yields the following definition for term frequencies:

$$tf(t,e) = \Big(\prod_{l \in path(e,se)} aw_l \Big) tf(t,se) \tag{7}$$

After having refined the definition of global and local statistics for flexible XML retrieval, the retrieval status value of an XML element e in the query scope is given by simply instantiating the $tf\,idf$ ranking function with the appropriate global and local statistics for tf and ief. Section 5 will explain how to compute statistics for combinations of several retrieval types in a query.

4 Storage Management with PowerDB-XML

Current approaches to XML processing have focused either on the data-centric or the document-centric side. One of the promises of XML however is to reconcile these – at least in practical settings. Therefore, the objective of the PowerDB-XML project at ETH Zurich is to support both data-centric and document-centric XML processing on a single integrated platform.

A straight-forward approach is to rely on relational database systems, to deploy data-centric database mapping techniques proposed by previous work, and to extend this setting with the functionality needed for document-centric processing. Several approaches have been pursued already to combine document-centric processing with database systems: most commercially available database systems for instance feature extensions for text processing. This enables retrieval over textual content stored in database table columns. However, this does not allow for flexible weighting granularities as discussed in Sect. 3. In particular, query-specific statistics according to the scope of the query are not feasible since IR statistics are not exposed by the text extenders. Therefore, text extenders are not a viable solution.

An alternative approach is to couple a database system for data-centric processing with an information retrieval systems for document-centric processing, as pursued e.g. by [27]. This approach however suffers from the same drawback as the one previously mentioned: IR statistics are hidden by the information retrieval system and query-specific statistics are not possible.

The third approach pursued in the PowerDB-XML project is instead to rely on relational database systems and to realize document-centric functionality on top of the database system with standard SQL. This approach is based on the observation from own previous work [20] and the work by Grossman et al. [18] that information retrieval using relational database systems for storage management is efficient. The advantage of this approach is that storing IR index data in the database makes IR statistics available for document-centric XML processing with query-specific statistics. The following paragraphs discuss how to combine data-centric and document-centric storage management on relational database systems and outline the approach taken in PowerDB-XML.

Data-Centric Storage Management. Regarding data-centric database mappings, PowerDB-XML supports the mapping schemes proposed in previous work as

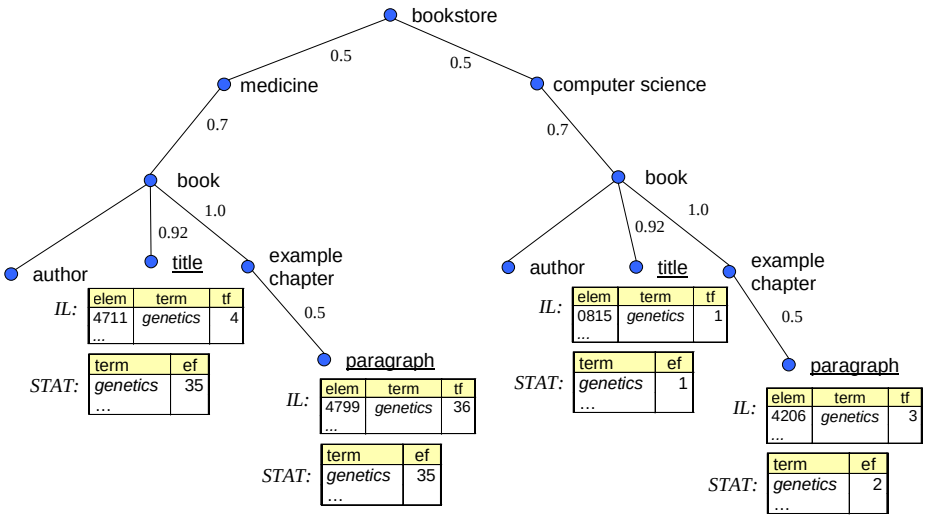

Fig. 2. Basic indexing nodes of the XML document in Fig. 1

discussed in Sect. 2. Our current implementation features text-based mappings, EDGE [7], and STORED [5]. An API allows users to define their own mappings and to deploy them to PowerDB-XML. An administrator then decides for a particular combination of mapping schemes that suits the XML applications running on top of PowerDB-XML.

Document-centric Storage Management. A naive solution to support flexible retrieval with query-specific statistics would be to keep indexes and statistics for each combination of element types and element nestings that could possibly occur in a query. However, the amount of storage that this approach requires for indexes and statistics is prohibitively large and is therefore not a viable solution. Hence, we refine the notion of indexing nodes as proposed by Fuhr et al. [13] to keep indexes and statistics only for basic element types. When it comes to single-category retrieval, multi-category retrieval or nested retrieval, the approach proposed here derives the required indexes and statistics from the underlying basic ones on-the-fly, i.e., at query runtime. This has the advantage that the amount of storage needed to process IR queries on XML content is small as compared to the naive approach.

To do so, flexible retrieval on XML documents first requires to identify the basic element types of an XML collection that contain textual content. These nodes are denoted as *basic indexing nodes*. There are several alternatives how to derive the basic indexing nodes from an XML collection:

– The decision can be taken completely automatically such that each distinct element type at the leaf level with textual content is treated as a separate indexing node.

– An alternative is that the user or an administrator decides how to assign element types to basic indexing nodes.

These approaches can further rely on an ontology that, for instance, suggests to group element types summary and abstract into the same basic indexing node. For ease of presentation, let us assume that the basic indexing nodes have already been determined, and the respective textual XML content already underwent IR pre-processing, including term extraction and stemming. PowerDB-XML then annotates the basic indexing nodes with the IR indexes and statistics derived from their textual content. Figure 2 illustrates this for the Data Guide [14, 15] of the example document in Figure 1. Element types with underlined names in the figure stand for basic indexing nodes and have been annotated with inverted list tables (*IL*) and statistic tables (*STAT*) for vector space retrieval. The *IL* tables store element identifiers, term occurrences, and local IR statistics (term frequencies for vector space retrieval) in the table columns `elem`, `term`, and `tf`, respectively. The global statistics tables *STAT* in turn store term identifiers and global statistics (element frequencies for vector space retrieval) in the table columns `term` and `ef`, respectively. PowerDB-XML keeps an *IL* and a *STAT* table for each leaf node of the collection structure that has textual content (cf. Fig. 2). The annotations of the edges in the figure represent augmentation weights.

5 Operators for Flexible XML Retrieval

5.1 Operators for Combined Retrieval Types

Depending on the scope of the IR query, a combination of single-category, multi-category, and nested retrieval may be necessary to compute the ranking. Depending on the retrieval granularity and the nesting, several inverted lists and statistics tables may be relevant. The following example illustrates this.

Example 4. Consider a nested retrieval request with the query scope //book and the query text 'XML Information Retrieval' on an XML collection like the one shown in Fig. 1. The request requires the functionality of nested retrieval since the examplechapter-sub-tree and the title sub-element are searched for relevant information. Moreover, the request also requires multi-category retrieval functionality since both computer science and medicine books may qualify for the result.

As the example shows, practical settings require a combination of the different retrieval types discussed in Section 3. In order to efficiently implement query processing for flexible IR on XML documents, this paper proposes operators called SINGLECAT, MULTICAT, NESTCAT, and AUG which encapsulate the functionality for integrating IR statistics and inverted lists. The results of a composition of these operators are integrated statistics and inverted lists for flexible retrieval from XML documents. The following paragraphs give an overview

about the operators and their signatures. Subsequent paragraphs in this section then discuss their implementation in more detail.

SINGLECAT returns the IR statistics and inverted list of a given basic indexing node under a particular query. SINGLECAT takes a path expression *expr* defining a basic indexing node and a set of query terms $\{term\}$ as input parameters. The signature of SINGLECAT is:

$$SINGLECAT(expr, \{term\}) \rightarrow (IL, STAT)$$

MULTICAT in turn takes several basic indexing nodes as input and integrates their statistics to the multi-category ones using Definition 5. MULTICAT has the following signature:

$$MULTICAT(\{(IL, STAT)\}) \rightarrow (IL, STAT)$$

NESTCAT computes the integrated IR statistics for sub-trees of XML collection structures. In contrast to MULTICAT, NESTCAT relies on Definition 3 and 7 to integrate statistics:

$$NESTCAT(\{(IL, STAT)\}) \rightarrow (IL, STAT)$$

Finally, the operator AUG downweighs term weights using the augmentation weights annotated to collection structure when propagating basic indexing node data upwards. The operator takes an inverted list and IR statistics as well as an augmentation weight *aw* as input parameters:

$$AUG((IL, STAT), aw) \rightarrow (IL, STAT)$$

The following example illustrates the combination of these operators in order to integrate inverted lists and IR statistics for flexible retrieval processing.

Example 5. Consider again the query 'XML Information Retrieval' on //book elements from the previous example in combination with the document collection underlying Figure 2. The operators SINGLECAT, MULTICAT, NESTCAT, and AUG integrate IR statistics and inverted lists as stored in the underlying basic indexing nodes for the scope according to the query. This yields the operator tree shown in Figure 3.

5.2 SQL-Implementation of Flexible Retrieval Operators

The following paragraphs explain in more detail how PowerDB-XML implements the operators for processing single-category retrieval, multi-category retrieval and nested retrieval using basic indexing nodes and standard SQL.

Single-Category Retrieval Processing. Combining different retrieval types in an XML request requires to make the basic indexing node information available for further processing. The SINGLECAT operator works on the global and local statistics of a basic indexing node. The following SQL code shows how PowerDB-XML implements the SINGLECAT operator on input tables *IL* and *STAT*.

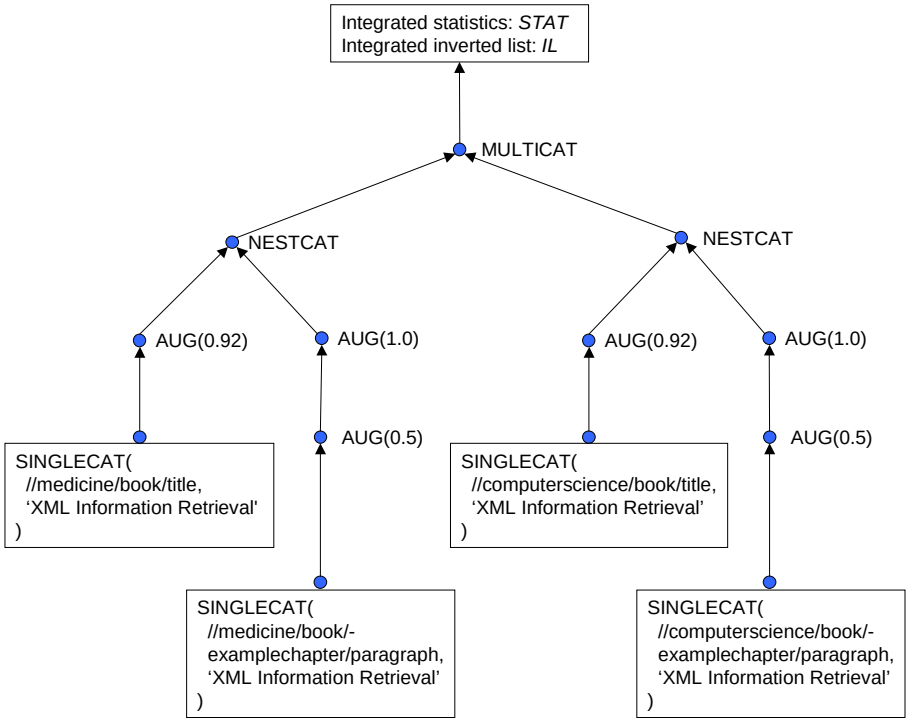

Fig. 3. Combination of retrieval types

```
SELECT i.elem, i.term, i.tf INTO IL′
FROM IL i, query q WHERE i.term = q.term

SELECT s.term, s.ef INTO STAT′
FROM STAT s, query q WHERE s.term = q.term
```

Multi-Category Retrieval Processing. Using basic indexing nodes directly for multi-category retrieval is not feasible since statistics are per basic indexing node. Hence, query processing must dynamically integrate the statistics when the query encompasses several categories.

MULTICAT relies on input provided by several – possibly parallel – invocations of the SINGLECAT operator. MULTICAT integrates their local and global statistics. Note that a simple set union suffices to integrate the postings to the inverted lists since they only carry local statistics such as term frequencies while global IR statistics such as element frequencies require integration using Definition 5. The following SQL code shows how PowerDB-XML implements the MULTICAT operator on input tables IL_1, $STAT_1$, IL_2, and $STAT_2$.

```
SELECT i.elem, i.term, i.tf INTO IL′
```

```
FROM IL i, query q WHERE i.term = q.term
SELECT i.elem, i.term, i.tf INTO IL'
FROM IL i, query q WHERE i.term = q.term

SELECT s.term, SUM(s.ef) INTO STAT'
FROM (SELECT * FROM STAT_1 UNION
      SELECT * FROM STAT_2) s
GROUP BY s.term
```

Nested Retrieval Processing. The operator NESTCAT implements the functionality for integrating local and global statistics for nested retrieval. In contrast to MULTICAT, simple set union for the inverted lists does not suffice with nested retrieval. Instead, an aggregation of the term frequencies (tf) in the XML subtrees is required (cf. Def. 7). Note that the tf values are assumed to be properly augmented by previous invocations of the AUG operator. Hence, a simple SQL SUM suffices to integrate the term frequencies. The following SQL code shows how PowerDB-XML implements the NESTCAT operator on input tables IL_1, $STAT_1$, IL_2, and $STAT_2$.

```
SELECT e.elem, i.term, SUM(i.tf) INTO IL'
FROM elements e, IL_1 i
WHERE DescendantOrSelf(e.elem, i_1.elem)
SELECT e.elem, i.term, SUM(i.tf) INTO IL'
FROM elements e, IL_2 i
WHERE DescendantOrSelf(e.elem, i_2.elem)

SELECT s.term, COUNT(DISTINCT i.elem) INTO STAT'
FROM IL'
GROUP BY i.term
```

Note that repeated application of the binary versions of the operators implements the n-ary ones.

Processing of Augmentation. The operator AUG implements augmentation weighting for flexible retrieval processing. As the following SQL code shows, it simply returns a weighted projection of the local statistics of the input table IL which correspond to term frequencies with vector space retrieval. Global statistics are not affected by augmentation. Hence, AUG simply propagates them without any changes to subsequent operator instances.

```
SELECT elem, term, aw * tf INTO IL'
FROM IL
```

Computing Retrieval Status Values with Query Specific Statistics. Previous work has proposed implementations using standard SQL for data access with Boolean, vector space and probabilistic retrieval models [10, 18]. Based on this work, the following SQL code for flexible XML retrieval using *tf idf* ranking takes integrated query-specific statistics *STAT* and *IL* from a composition of the operators discussed previously as an input.

```
SELECT elem, SUM(i.tf * ief(s.ef) * ief(s.ef) * q.tf) rsv
FROM IL i, STAT s, query q
WHERE i.term = s. term AND s.term = q.term
GROUP BY elem
```

The SQL statement yields the ranking, i.e., the XML element identifiers from the query scope and their retrieval status values.

Preliminary Experimental Results with INEX 2002. INEX – short for the INitiative for the Evaluation of XML retrieval – is an ongoing effort to benchmark the retrieval quality of XML retrieval systems [12]. INEX comes with a document collection of roughly 500 MB of XML documents representing about 12,000 IEEE Computer Society publications. Marked up in XML, the document collection comprises about 18.5 million XML elements. 60 different topics have been developed by the initiative, including relevance assessments. INEX differentiates between so-called *content-only (CO) topics* and *content-and-structure (CAS) topics*. CO topics specify a query text or a set of keywords for relevance-oriented search. Hence, each of the 18.5 million XML elements in the collection is a potential results to a CO topic. CAS topics in addition pose structural constraints such as path expressions on the result elements.

Using PowerDB-XML as retrieval engine, we have run the INEX 2002 benchmark on a single PC node with one 1.8 GHz Pentium processor, 512 MB RAM, and a 40 GB IDE disk drive. We deploy Microsoft Windows 2000 Server as operating system and Microsoft SQL Server 2000 for storage management. After having loaded PowerDB-XML with the complete document collection, we have run all topics and measured their response times. A positive finding from this series of experiments is that CAS topic processing is interactive, i.e., response times are in the order of seconds. However, some CO topics yield response times in the order of minutes (but less than 10 minutes). The reason for this is that CO topics require to compute a ranking for potentially all 18.5 million XML elements since constraints on the document structure to cut down the result space are not available with this topic type. The bottleneck of the topics with high response times is the inverted list lookup of terms in combination with processing element containment relationships. Note that the overhead of computing and integrating global statistics such as *ief* values is not significant with both topic types. We therefore plan to investigate combining query-specific statistics with more efficient representations of element containment relationships in relational database systems as discussed, e.g., in [19].

6 Conclusions

So far, database research has focused on data-centric processing of XML documents. This has left aside a large number of XML applications which require document-centric XML processing. An explanation for this might be that document-centric processing and in particular ranked and weighted retrieval from XML content is still an open research issue in the IR community. Our work on PowerDB-XML took over the vector-space model and $tf\,idf$ ranking from flat document retrieval and refined it to flexible retrieval on XML documents using query-specific statistics. We see this as an important foundation for document-centric XML processing and we are currently evaluating its retrieval quality in the INEX initiative [11].

As our discussion has shown, flexible XML retrieval with query-specific statistics nicely maps to relational storage managers and an efficient implementation with standard SQL. This allows to combine state-of-the-art data-centric database mapping schemes with our relational implementation of document-centric XML processing. We pursue this approach in the PowerDB-XML project at ETH Zurich. As a result, our XML engine covers the full range from data-centric to document-centric XML applications on a single integrated platform using XML for data representation.

References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web – From Relations to Semistructured Data and XML.* Morgan Kaufmann Publishers, 2000.
2. P. Bohannon, J. Freire, J. R. Haritsa, M. Ramanath, P. Roy, and J. Simon. LegoDB: Customizing Relational Storage for XML Documents. In *Proceedings of 28th International Conference on Very Large Data Bases (VLDB2002), August 20-23, 2002, Hongkong, China*, pages 1091–1094. Morgan Kaufmann, 2002.
3. P. Bohannon, J. Freire, P. Roy, and J. Simon. From XML Schema to Relations: A Cost-based Approach to XML Storage. In *Proceedings of the 18th International Conference on Data Engineering (ICDE2002), February 26 - March 1, 2002, San Jose, CA, USA*. Morgan Kaufmann, 2002.
4. M. J. Carey, J. Kiernan, J. Shanmugasundaram, E. J. Shekita, and S. N. Subramanian. XPERANTO: Middleware for Publishing Object-Relational Data as XML Documents. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *Proceedings of 26th International Conference on Very Large Data Bases (VLDB2000), September 10-14, 2000, Cairo, Egypt*, pages 646–648. Morgan Kaufmann, 2000.
5. A. Deutsch, M. F. Fernandez, and D. Suciu. Storing Semistructured Data with STORED. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 431–442. ACM Press, 1999.
6. M. F. Fernandez, W. C. Tan, and D. Suciu. SilkRoute: Trading between Relations and XML. *WWW9 / Computer Networks*, 33(1-6):723–745, 2000.
7. D. Florescu and D. Kossmann. Storing and Querying XML Data using an RDMBS. *IEEE Data Engineering Bulletin*, 22(3):27–34, 1999.

8. D. Florescu, D. Kossmann, and I. Manolescu. Integrating Keyword Search into XML Query Processing. In *Proceedings of the International WWW Conference, Amsterdam, May 2000*. Elsevier, 2000.
9. E. Fox and M. Koll. Practical Enhanced Boolean Retrieval: Experiments with the SMART and SIRE Systems. *Information Processing and Management*, 24(3):257–267, 1988.
10. O. Frieder, A. Chowdhury, D. Grossman, and M. McCabe. On the Integration of Structured Data and Text: A Review of the SIRE Architecture. In *Proceedings of the First DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries, Zurich, Switzerland, 2000*, pages 53–58. ERCIM, 2000.
11. N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas. INEX: Initiative for the Evaluation of XML Retrieval. In R. Baeza-Yates, N. Fuhr, and Y. S. Maarek, editors, *Proceedings of the ACM SIGIR Workshop on XML and Information Retrieval, Tampere, Finland*, pages 62–70. ACM Press, 2002.
12. N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas, editors. *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX), 9-11 December 2002, Schloss Dagstuhl, Germany*. ERCIM DELOS, 2003. ERCIM-03-W03.
13. N. Fuhr and K. Großjohann. XIRQL: A Query Language for Information Retrieval in XML Documents. In W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, editors, *Proceedings of the 24th Annual ACM SIGIR Conference on Research and Development in Information Retrieval, New Orleans, USA*, pages 172–180. ACM Press, 2001.
14. R. Goldman, J. McHugh, and J. Widom. From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. In *ACM SIGMOD Workshop on The Web and Databases (WebDB'99), June 3-4, 1999, Philadelphia, Pennsylvania, USA*, pages 25–30. INRIA, Informal Proceedings, 1999.
15. R. Goldman and J. Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *Proceedings of 23rd International Conference on Very Large Data Bases, 1997, Athens, Greece*, pages 436–445. Morgan Kaufmann, 1997.
16. T. Grabs, K. Böhm, and H.-J. Schek. PowerDB-IR – Information Retrieval on Top of a Database Cluster. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM2001), November 5-10, 2001 Atlanta, GA, USA*, pages 411–418. ACM Press, 2001.
17. T. Grabs and H.-J. Schek. Generating Vector Spaces On-the-fly for Flexible XML Retrieval. In R. Baeza-Yates, N. Fuhr, and Y. S. Maarek, editors, *Proceedings of the ACM SIGIR Workshop on XML and Information Retrieval, Tampere, Finland*, pages 4–13. ACM Press, 2002.
18. D. A. Grossman, O. Frieder, D. O. Holmes, and D. C. Roberts. Integrating Structured Data and Text: A Relational Approach. *Journal of the American Society for Information Science (JASIS)*, 48(2):122–132, Feb. 1997.
19. L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. Xrank: Ranked keyword search over xml documents. In A. Y. Halevy, Z. G. Ives, and A. Doan, editors, *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, pages 16–27. ACM, 2003.
20. H. Kaufmann and H. J. Schek. Text Search Using Database Systems Revisited - Some Experiments -. In *Proceedings of the 13th British National Conference on Databases*, pages 18–20, 1995.

21. M. Rys. Bringing the Internet to Your Database: Using SQLServer 2000 and XML to Build Loosely-Coupled Systems. In *Proceedings of the 17th International Conference on Data Engineering, 2001, Heidelberg, Germany*, pages 465–472. IEEE Computer Society, 2001.
22. G. Salton. *The SMART Retrieval System : Experiments in Automatic Document Processing*. Prentice-Hall, 1971.
23. G. Salton, E. A. Fox, and H. Wu. Extended Boolean Information Retrieval. *Commun. ACM*, 26(12):1022–1036, 1983.
24. G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
25. J. Shanmugasundaram, J. Kiernan, E. J. Shekita, C. Fan, and J. Funderburk. Querying XML Views of Relational Data. In P. M. G. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. T. Snodgrass, editors, *Proceedings of 27th International Conference on Very Large Data Bases, September, 2001, Roma, Italy*, pages 261–270. Morgan Kaufmann, 2001.
26. J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, and J. Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. In M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, editors, *Proceedings of 25th International Conference on Very Large Data Bases (VLDB'99), September 7-10, 1999, Edinburgh, Scotland, UK*, pages 302–314. Morgan Kaufmann, 1999.
27. M. Volz, K. Aberer, and K. Böhm. Applying a Flexible OODBMS-IRS-Coupling for Structured Document Handling. In S. Y. W. Su, editor, *Proceedings of the Twelfth International Conference on Data Engineering, New Orleans, Louisiana, USA*, pages 10–19. IEEE Computer Society, 1996.
28. W3C – World Wide Web Consortium (J. Clark, S. DeRose, editors). XML Path Language (XPath) Version 1.0. `http://www.w3.org/TR/xpath`, Nov. 1999.
29. W3C – World Wide Web Consortium (S. Boag, D. Chamberlin, M. F. Fernandez, D. Florescu, J. Robie, J. Simon, editors). XQuery 1.0: An XML Query Language. `http://www.w3.org/TR/xquery`, Nov. 2002.
30. W3C – World Wide Web Consortium (T. Bray, J. Paoli, C. M. Sperberg-McQueen, editors). Extensible Markup Language (XML) 1.0. `http://www.w3.org/TR/1998/REC-xml-19980210`, Feb. 1998.