Schema Extraction for XML Document Retrieval

Jong P. Yoon
Center for Advanced Computer Studies
University of Louisiana, Lafayette
Lafayette, LA 70504-4330
jyoon@cacs.usl.edu

Sungrim Kim
Department of Computer Science
Sookmyung W. University
Seoul, 710-140 Korea
srkim@cs.sookmyung.ac.kr

Abstract
An XML document is a collection of multiple types of data sets tagged by XML elements. Such an XML document can be retrieved not only by a Boolean connection with keywords but also by XML element-based query languages. In many cases, however, keywords-based queries result in either too many hits or too few results. It is not trivial to formulate what to retrieve a “good” sized query-result. This paper proposes a method of schema extraction for multimedia XML document collection. Schemas are then levelized with respect to the frequency of topological document structures in a database. The topological structural information of these schemas is used to formulate queries and further to rewrite queries for relaxation and restriction. Without modification, the method proposed in this paper is used not only for multimedia XML document collections but for general XML databases.

Keywords: Bitmap Approach to Schema Extraction, Multi-level XML Schemas, Query Rewriting

1. Introduction
EXtensible Markup Language (XML) [Bray (1998)] is a standard for data representation and exchange on the Internet. It is possible that multimedia data is represented and integrated in XML. In the Internet, an XML-element tagged text data, which is semistructured, is able to integrated with a multimedia data that is usually unstructured (e.g., Internet news or Internet gallery). We call such a data multimedia XML document. Because of a semistructured XML text data can be integrated with an unstructured multimedia data, querying against multimedia XML documents is not trivial. Without knowing the schema for multimedia XML documents, it is not easy to formulate what to exactly specify when one wants to retrieve them.

1.1. Motivating Examples

EXAMPLE 1.1 An example of semantically complex queries is to “retrieve all documents that describe about computers and contain a computer image.” If such a data is XML-element tagged in a database (or on the WWW), and if the schema information is provided to users, the example query may be formulated as follows: “retrieve all documents that describe computers in a paragraph, and contain computers in the caption of computer image.”

EXAMPLE 1.2 Schemas extracted for multimedia XML documents is also useful for query optimization. In the previous example, suppose that there are too few results matched. However, if yet another keyboard, monitor, and main unit are separately available, we may collect them to show. The original query may be rewritten as follows: “retrieve all documents that describe keyboards, monitors, or main units in the caption of images and the image of those three items.” Queries can be rewritten to retrieve “good” sized results by levelizing up or down schemas.
1.2. Related Work

With the recent emergence of XML [Bray (1998)], a proposed standard for exchanging information on the Web [Light (1997)], and the remarkable similarity of XML to typical models for semistructured data, support for query languages for semistructured data – and the performance of such queries over large semistructured databases – is of increasing importance.

The extraction of the schema from a semistructured documents follows either unsupervised categorization or supervised categorization. The former we call *clustering*, the latter *classification*. Many researchers focus on classification of semistructured data or documents [Chakrabarti (1998), Florescu (1997)]. One example of classification is summarization of documents [Dolin (1999)]. The approach presented in this paper is instead to “cluster” documents. In general, semistructured documents are clustered according to XML-elements. After a schema implicit in the cluster of semistructured documents is extracted, abstraction techniques are applied to the initial schema.

A *generalized path expression*, useful in the context of XML-like semistructured databases, allows label wildcards and regular expression operators [Abiteboul (1997), Fernandez (1998)]. Generalized path expression optimization has been studied in [Christophides (1996), Fernandez (1998B)]. [Fernandez (1998B)] describes a query rewrite technique that transforms generalized path expressions to simpler forms prior to optimization. In [Christophides (1996)], an algebraic optimization framework is proposed specifically to avoid exponential blow-up in the presence of closure operators. Our work is similar in spirit, but not in details, to [Fernandez (1998B)]. In [Fernandez (1998B)], a cross-product is computed between a *graph schema* – a summary of the database that must be small and reside in memory – and a representation of the query. The work in this paper, however, describes not only generalization of label paths but also aggregation and redundancy elimination of them. In addition, this paper deals with a way of integration of levelized schemas in order to guide users to formulate queries and also to make it possible to process the queries efficiently.

1.3. Organization

Remaining of this paper is organized as follows. Section 2 describes preliminaries. Section 3 describes methods of multi-level schema extraction. Section 4 describes a query rewriting method using multi-level schemas and the measurements of rewritten queries in terms of coverage and accuracy. Section 5 concludes this paper.

2. Preliminaries

2.1. Document Type Definition and Instances

XML provides a simple and general markup facility that is useful to represent a complex multimedia data together with a text data. Such a representation is possible by using a Document Type Definition (DTD). A DTD defines a class of XML data using a language that is essentially a context free grammar with several restrictions. For example, one may use the DTD declaration to constrain multimedia XML documents as shown in Figure 1.

Notice that XML DTD follows notational convention, such as ?, * and + denoting respectively zero or one, zero or more, and one or more occurrences of the preceding construct. We explain part of the declaration of the DTD in Figure 1. We assume that the types of all the other elements are PCDATA unless otherwise specified in the DTD. In the line (1), the mmdoc element contains one title element, one or more author elements, zero or more related-story elements and one body element, and has one optional date element. The element mmdoc has also an attribute called id. In the line (2), the author element contains the elements lastname and firstname together, or the single element fullname. XML multimedia data often specifies nested and cyclic structures, such as trees, directed graphs, and arbitrary graphs. In.
the line (3), the related-story element is again the element mmdoc. The attribute src of the element may have a value from the domain of mmdoc’s id. Notice this constraint does not appear in the DTD. In the line (4), the body element contains the elements para and optionally the element image. The para element contains the elements lang and text in the line (5). The lang element may have two attributes: 1) code character data type with default value ASCII, and 2) three possible sources of English (default value), Spanish, and French. The text element in the line (7) has two attributes: 1) href hyperlink reference, and 2) three possible region of left, center, and right. The image element contains caption with the component image or the img link in the line (8). In this description, an image can contain one or more images within itself. That is, an image multimedia data is composed with one or more component images. The img element contains the attributes src, height and width of an image in the line (9).

Multimedia XML document instances are now created and tagged according to the above DTD. Two document examples of them appear in Figure 2: one document from the line (1) to the line (16), and another document from (17) to (32). Notice that the image “Jefferson” is composed with the three component images like ”statue”, “constitution” and ”memorial-dome” as shown in the line (23) to the line (26).

2.2. Data Graph

Multimedia XML data instances, like semistructured data [Goldman (1997)], can be thought of as a labeled directed graph. The data defined in the previous subsection can be depicted as given in Figure 3. We call this a data graph. The nodes in the graph are objects; each object has a unique object identifier (oid), such as &01. Atomic objects have no outgoing edges and contain a value from one of the basic atomic types such as integer, string, gif, video, etc. All other objects may have outgoing edges and are called complex objects. Object &04 is complex and its subobjects are &07 and &08. Object &07 and &08 are atomic objects and have values, say “Henry” and “Smith.” There is a label to an arc between objects. For example, the label to the arc from the object &04 to the object &07 is called “lastname.”

2.3. Label Path Expression

In the data graph, there exists a path from the root node to a sub-node. A path is expressed in terms of labels. This path is called label path expression as in [Abiteboul (1997)]. A label path is expressed as a sequence of $p_i = l_1, l_2, l_3, \ldots, l_n$, where $l_j$ denotes a label for an arc from an object $x$ to its adjacent subobject $y$. $p_i$ is an outer tag of $p_j$ if $p_i$ is a prefix part in $p_j$. For example, in Figure 4, one of the label path expressions is mmdoc.author.lastname

```
(1)<!ELEMENT mmdoc (title, author+, related-story*, body, date?)>
<!ATTLIST mmdoc id CDATA #REQUIRED>
(2)<!ELEMENT author ((lastname,firstname)| fullname)>
(3)<!ELEMENT related-story (#PCDATA)>
<!ATTLIST related-story src CDATA #REQUIRED>
(4)<!ELEMENT body (para, image?)+>
(5)<!ELEMENT para (#PCDATA|(lang?,text)+)>
(6)<!ELEMENT lang (#PCDATA)>
<!ATTLIST lang code CDATA "ASCII" source (eng|spn|frn) eng>
(7)<!ELEMENT text (#PCDATA)>
<!ATTLIST text href CDATA #REQUIRED
region (left|center|right) left>
(8)<!ELEMENT image (caption?, (image|img))>
(9)<!ELEMENT img (#PCDATA)>
<!ATTLIST img src CDATA #REQUIRED
height CDATA #IMPLIED
width CDATA #IMPLIED>
```

Figure 1. DTD for Multimedia Documents
2.4. Lattice

Suppose we are given a set of XML-elements $L$, and a partial ordering $\prec$ on the elements in $L$. Consider two labels $l_1$ and $l_2$. When $l_1 \prec l_2$, we say that $l_2$ is a more general label than $l_1$. We call $l_2$ a ancestor of $l_1$ if $l_1 \prec l_2$. When $l_1 \prec l_2$, we say that $l_1$ is a more specific label than $l_2$. We call $l_1$ an descendant of $l_2$ if $l_1 \prec l_2$. Notice that the lattice can be depicted upside down to be like a schema graph in this paper. For example, in Figure 4, the label `mmdoc.author.lastname` can be obtained using only the values bound to the label `mmdoc.author`. Thus `mmdoc.author.lastname` $\prec$ `mmdoc.author`.

Note that $\prec$ imposes a partial ordering on the label expression, and it is transitive. We shall talk about the relationships of label expression as forming a lattice [Tremblay (1975)]. In order to be a lattice, any two elements (i.e., tagged elements or label) must have a least upper bound and a greatest lower bound according to the $\prec$ ordering. However, in practice, we only need the assumptions that

1. $\prec$ is a partial order, and
2. There is a top () element, a label upon which every label is dependent.

The greatest lower bound of labels $l_1$ and $l_2$, denoted by $\text{glb}(l_1, l_2)$, is the least common ancestor of $l_1$ and $l_2$ if one exists. The least upper bound of labels $l_1$ and $l_2$, denoted by $\text{lub}(l_1, l_2)$, is the least common descendant of $l_1$ and $l_2$ if one exists. For example, $\text{glb}(&64, &70) = &40$ and $\text{lub}(&23, &42) = &56$ in Figure 3.

2.5. Querying XML Data

Figure 2. Multimedia XML Document Examples (stored in the file www.a.b.e/mmdocument.xml)
Research on semistructured data has addressed query-language design [Abiteboul (1997), Buneman (1997), Deutsch (1999), Fernandez (1998A)], and query processing and optimization [McHugh (1999)]. The web site http://www.w3.org/TandS/XML/QL98/ contains a more complete list of query languages. In this section, we use the XML-QL language [Deutsch (1999)], as an example language among many, to query multimedia data and further to rewrite a user query.

The example below selects all image data with the title authored by Smith in the year later than 1990 if the image or any component image is described with the word "rock." XML-QL queries consists of a WHERE clause, specifying what to select, and a CONSTRUCT clause, specifying what to return.

WHERE <mmdoc> <author> <lastname>Smith </> </> <title> $t </> <body><image><image>$i<caption>$d</></></>}</><date> $y </> IN "www.a.b.e/mmdocument.xml", (contains, $d, "rock"), $y>1990

CONSTRUCT <result> <title> $t </> <body> <image> <image> $i </></></>}</

The expression <mmdoc> ... </mmdoc> in the WHERE clause is called a condition pattern, where the one in the CONSTRUCT clause is called a display pattern. Note that contains in the WHERE clause is a user-defined function, specifying that $d contains the term "rock". In the later section, the condition pattern and display pattern will be rewritten to retrieve a "good" sized XML document.

3. Schema Extraction

In this section, we describe a conventional schema for a XML DTD and for XML instances. Between
employ the bitmap-based operations.

3.1. Initial Extraction for DTD

We can construct an XML DTD graph using the same manner in Section 2.2. The XML Data graph for the mmdoc DTD shown in Figure 1 is in Figure 4. The DTD represents a context-free grammar for XML data instances. We think that the data graph of the DTD is the most abstracted schema for those XML data instances. We call such a DTD graph one of schema graphs as opposed to the data graph.

3.2. Initial Extraction for XML Instances

A data schema is defined to be a concise, accurate, and convenient summary of the structure of a database. We first employ the concept of a so-called DataGuide schema, in Lore [Goldman (1997)]. The schemas presented in this paper can be depicted in a directed graph as data are depicted. To achieve conciseness, Lore specifies that a DataGuide describes every unique label path of data instances exactly once, regardless of the number of times it appears in those data instances. To ensure accuracy, Lore specifies that the DataGuide encodes no label path that does not appear in the data instances. For convenience, a DataGuide itself is an object so the object can be stored and accessed by using the same techniques available for processing typical databases. One of the schemas is shown in Figure 5.

We argue that although the schema described by [Goldman (1997)] is complete enough, it is not yet concise. The DTD is more concise enough, but it does not represent the structural information about data instances. Between these two extreme ends of schemas, there are many gray-levels of schemas. We investigate that there may be possible two approaches to gray-level schema extractions for multimedia XML data: 1) Topological structure based schema extraction, and 2) Frequent structure based schema extraction. The former approach is described in Section 3.3. Section 3.4 describes a bitmap approach of the frequency structure based schema extraction.

3.3. Topological-Structure based Schema Extraction

3.3.1. Schema Generalization from Initial Schema

Each label $l$ of an object $x$ may have a set of subobjects of $x$. We denote the set of subobjects of $x$ by DOM $(x.l)$. The generalization operation is a partial function from DOM $(x.l_1)$ to DOM $(y.l_2)$, denoted $x.l_1 \xrightarrow{\text{generalize}} y.l_2$, where $l_1 < l_2$ and DOM $(x.l_1) \subseteq$ DOM $(y.l_2)$. That is, two labels $x.l_1$ and $y.l_2$ are generalized to the label $y.l_2$ because $(y.l_2) = \text{glb}(x.l_1,y.l_2)$. For example in Figure 5, there is a partial order mmdoc.body.para.text $\prec$ mmdoc.body.para. We then see that mmdoc.body.para.text

![Figure 4. XML-Schema Graph of mmdoc XML DTD](image)


is generalized to mmdoc.body.para because mmdoc.body.para = llb(mmdoc.body.para, mmdoc.body.para.text) and \( \text{DOM} (\text{mmdoc.body.para.text}) \subseteq \text{DOM} (\text{mmdoc.body.para}) \).

3.3.2. Schema Aggregation from Initial Schema

Each label \( l \) of an object \( x \) has associated with it a set of subobjects of \( x \). The aggregation operation is a partial function from \( \text{DOM} (x.l_1) \) to \( \text{DOM} (x.l_2) \), denoted \( x.l_1 \Rightarrow_{\text{aggregate}} x.l_2 \), where \( l_1 \prec l_2 \) and \( \text{DOM} (x.l_2) = \text{DOM} (x.l_1) \cup \text{DOM} (x.l_2) \). In Figure 5, for example, we see that mmdoc.body.image.image is aggregated to mmdoc.body.image because mmdoc.body.image.image is aggregated to mmdoc.body.image and \( \text{DOM} (\text{mmdoc.body.image}) = \text{DOM} (\text{mmdoc.body.image.image}) \cup \text{DOM} (\text{mmdoc.body.image}) \).

3.4. Bitmap Approach to XML Schema Extraction

We employ bitmap-indexing [Chan (1999)] to store the information of label path expressions of XML documents, and to exploit for query matching and for schema abstraction. The idea of using a bitmap index is to replace the label path expression with a bit vector (or bitmap). Each bitmap represents the occurrence of labels for an XML document. That is, a bitmap index is essentially a collection of bitmaps. The size of bitmap is equal to the cardinality of indexed documents, and its dimensionality is the number of the labels in data graphs. In the simplest bitmap index design, the \( i \)th bit of a bitmap associated with label path expression \( p_i \) is set to 1 if and only if the \( i \)th document has the path \( p_i \).

As an example, the following bitmap index shows a part of the labels in Figure 3. Let \( d \) denote an XML document, and \( D \) be a document collection. Let \( p_1 \) denote the label path expression mmdoc.title, and \( p_2 \) denote mmdoc.author, etc. A major advantage of bitmap indexes is that bitmap manipulations using bitwise operators (AND, OR, XOR, NOT) are very efficiently supported by hardware.

![Figure 5. XML-Schema Graph of mmdoc XML Documents](image-url)
We first define the distance between two multimedia XML data sets as follows:

- **Distance of two MXDs.** The distance of two MXDs $d_i$ and $d_j$ is defined as the number of 1’s in XOR ($d_i, d_j$), denoted $|\text{XOR}(d_i, d_j)|$.

- **Normalized Distance of two MXD’s.** The normalized distance of two MXDs $d_i$ and $d_j$ is defined as $\frac{|\text{XOR}(d_i, d_j)|}{\max(|d_i|,|d_j|)}$.

Using this distance definition, we can determine the similarity between two XML documents.

- **Similarity.** Two XML documents $d_i$ and $d_j$ are similar if $\frac{|\text{XOR}(d_i, d_j)|}{\max(|d_i|,|d_j|)} \leq \epsilon$, for a given threshold $\epsilon$.

We propose the schema extraction rules (some are below). Let $|p|$ be the number of multimedia XML data from the binary bitwise operation AND(path_mask of $p_i$, $D$). Let $|d|$ be the total number of multimedia XML data collections. A path_mask of $p_i$, e.g., 000000010...0 for $p_8$, is a bitmap by setting the $i$th bit to 1 and all other bits to 0 in a unit bitmap.

- For any two multimedia XML documents, $d_i$ and $d_j$, if AND($d_i$, $d_j$) is the same as $d_i$ and $d_j$, then $d_i$ and $d_j$ have exactly the same tags. For example, $d_3$ and $d_5$ have the same tags.

- If the occurrence rate $\frac{|\text{AND}(D, \text{path_mask of } p_i)|}{|D|} \geq \epsilon$, then $p_i$ appears in the schema, where a threshold is given from $0 \leq \epsilon \leq 1$. Otherwise, the path does not appear in the schema. For example, the label path expressions mmdoc.body.para.text.href and mmdoc.body.para.text.region do not appear in Figure 6(d) due to the occurrence rate of that path expression is below 0.2. If $\epsilon = 0$, then our approach is the same as Lore [Goldman (1997)].

- For any two paths $p_i$ and $p_n$, $p_n$ can be removed from the schema, if (1) $p_i$ is an outer tag of $p_n$, (2) $p_i$ has repeated labels as a suffix, and (3) $\frac{|\text{AND}(D, \text{path_mask of } p_i)|}{|\text{AND}(D, \text{path_mask of } p_n)|} \geq \epsilon$. As an example, the path label mmdoc.body.para appears instead of having mmdoc.body.para.text and mmdoc.body.para.lang in Figure 6(b). The schema at level = 0.05 does not contain the labels mmdoc.body.para.text and mmdoc.body.para.lang.

The schemas extracted by using the above rules are more flexible and available in multiple levels: from the most concise ($\epsilon = 1$) to the most complete or accurate ($\epsilon = 0$) levels. An example of a concise level schema is in Figure 6.

### 3.5 Multi-level XML Schemas

The user-specified threshold ($0 \leq \epsilon \leq 1$) dominates the levels of schemas that are extracted from multimedia XML data instances. The granularity of schema levelization takes user-specified threshold into account. According to user-specified threshold, we may have different schemas, and those schemas are then used to optimize user-specified queries. A set of multi-level XML schema is defined as $S = \{S_i|S_i$ is a schema at the level $i = \epsilon \}$, where $i$ is ordered according to user-specified thresholds.

For example, multi-level schemas are shown in Figure 6 which can be used for query formulation and optimization. The multi-level XML schemas are in the set $S = \{S_{0.2}, S_{0.15}, S_{0.1}, S_{0.05}\}$. 
Figure 6. Multi-level XML Schemas
4. Query Rewriting

4.1 Query Rewriting Method

We focus on rewriting a query condition. The condition part of a query appears in the WHERE clause. Querying multimedia XML data may result in too many answers or too few answers. If a user is not satisfied with the answers, we may want to relax or restrict the given query. There are basically three methods handling query relaxation problems.

- Naive approach: Query result is given to users without rewriting the query. No further processing is conducted.
- User-interactive approach: The system at least knows the quantity of results and shows it to users. Users are allowed to rewrite a query according to a system-given guideline.
- Query optimization: The system knows the problem and rewrites a query according to the user-specified level of granularity. In this section, we propose the third approach.

If a result to a query is too many, then the query can be rewritten in the schema whose level is greater than the original. This query rewriting is to restrict queries, and as such a “good” sized result will be produced. Query rewriting for query restriction can be conducted by taking lub() into account. Practically speaking, in multi-level XML schemas, a schema with a lower level can be used for query restriction.

On the other hand if a result to a query is too few, then the query is rewritten in the schema whose level is smaller than the original. This query rewriting is to relax queries, and therefore a “good” sized result will be produced. Query rewriting for query relaxation can be done by taking glb() into account. In multi-level XML schemas again, a schema with a higher level can be used for query relaxation.

For example, the following query is formulated in the schema (S 0.1) at the level \( \varepsilon = 0.1 \) in Figure 6. The query requests the component images of image data whose author is Smith if the caption of the component images contains the word “rock” and the date is later than 1990.

WHERE <mmdoc> <author> <lastname>Smith </> </>
<title> $t $</title>
<body><image><image>$i $</image></>
<caption>$d $</caption>
<date> $y $</date>
</mmdoc> IN "www.a.b.e/mmdocument.xml",
(contains, $d, "rock"), $y>1990

CONSTRUCT <result> <title> $t $</title>
<body> <image><image>
<img> $i $</image></> </body> </>
</>

Suppose that the query condition pattern over the node mmdoc.body.image.image.caption is matched with a few XML document instances. In order to retrieve more answers, the query condition pattern is relaxed by using the schema (S0.15) at the level \( \varepsilon = 0.15 \). A rewritten query is as follows:

WHERE <mmdoc> <author> <lastname>Smith </> </>
<title> $t $</title>
<body><image><image>$i $</image></>
<caption>$d $</caption>
<date> $y $</date>
</mmdoc> IN "www.a.b.e/mmdocument.xml",
(contains, $d, "rock"), $y>1990

CONSTRUCT <result><title> $t $</title>
<body><image><img>$i</image></>
</>
</>

As opposed to the original query, the above rewritten query requests all the image data, with ignorance of the component images, the documents authored by Smith in the year later than 1990.

We develop an algorithm to relax a label that is used in a query. The algorithm takes a schema and a label. That label is taken from the schema to formulate a query. The algorithm returns a new label which is relaxed from the original label and therefore can be matched with more data.
Also, in query restriction, lemmas 6.1 and 6.2 are switched in a sense that The original query. Those techniques could be categorized into three types, the results generated from applying the original query \( \sigma_{Q}(X) \cap \sigma_{D}(X) \) and the common results.

**Lemma 6.2:** Let \( Q_{i}, Q_{i+1}, \ldots, Q_{r-1} \) be a set of relaxed queries generated by the Relaxation procedure for a query \( Q \). For any two relaxed queries \( Q_{i} \) and \( Q_{j} \), \( 1 \leq i < j \leq r - 1 \), \( \text{Accuracy}(Q_{i}, Q_{j}) > \text{Accuracy}(Q_{i}, Q_{j}) \).

**Proof:** Following the Relaxation procedure, \( \sigma_{Q}(X) \) is always included in \( \sigma_{D}(X) \), i.e., 
\[
\frac{|\sigma_{Q}(X) \cap \sigma_{D}(X)|}{|\sigma_{Q}(X)|} \geq 1.
\]

Lemma 6.2: Let \( Q_{0}, Q_{1}, \ldots, Q_{r-1} \) be a set of relaxed queries generated by the Relaxation procedure for a query \( Q \). For any two relaxed queries \( Q_{i} \) and \( Q_{j} \), \( 1 \leq i < j \leq r - 1 \), \( \text{Accuracy}(Q_{i}, Q_{j}) > \text{Accuracy}(Q_{i}, Q_{j}) \).

**Proof:** Following the Relaxation procedure, \( \sigma_{Q}(X) \) is always included in \( \sigma_{D}(X) \), i.e., 
\[
\frac{|\sigma_{Q}(X) \cap \sigma_{D}(X)|}{|\sigma_{Q}(X)|} \geq 1.
\]

In query restriction, the inclusion and expansion concepts used in the query relaxation could be reversed and applied. For example, inclusion concept in query restriction could be applied by changing \( A \lor B \) to \( A \) and then to \( A \land B \). Also, in query restriction, lemmas 6.1 and 6.2 are switched in a sense that The Restriction procedure always generates restricted queries with \( \text{Accuracy} = 1 \), and for any two restricted queries \( Q_{i} \) and \( Q_{j} \), \( 1 \leq i < j \leq r - 1 \), \( \text{Coverage}(Q_{i}, Q_{j}) > \text{Coverage}(Q_{i}, Q_{j}) \).

Several techniques could be used to calculate \( \text{Accuracy} \) values for the relaxed queries with respect to the original query. Those techniques could be categorized into three types,

- Use estimated values of elements selectivities and dependencies to reflect the workload of the XML document repository.
• Apply the original and relaxed queries on the actual XML document repository.
• Use some estimated values of elements selectivities and dependencies along with applying selected queries on the actual XML document repository.

In each of the above general categories, Accuracy values of estimated queries are generated.

5. Conclusion

In this paper, we proposed how XML data can be represented with multimedia data, and how multimedia data can be retrieved by using XML-QL. Schemas can be extracted for multimedia XML document collections. An extracted schema can be generalized at each level of user-given thresholds. For a multiple level, we used a bitmap approach to the extraction of multi-level schemas for XML documents. These multi-level schemas are useful for query optimization. Given an XML-QL, we examined that the user query condition patterns are rewritten for intelligent retrieval. A query is rewritten by relaxing and restricting the labels in the WHERE clause.

Acknowledgement

This work was in part supported by LEQSF research grant. The authors would like to thank V. Raghavan, A. Hafez, and B. Kim for useful discussions and help with the query measurements.

References

Large Data Bases, pages 93–100, 1999.

