Advances
In Multimedia
Information Systems

6th International Workshop
on Multimedia Information Systems (MIS’00)
Proceedings

Walid G. Aref
Ibrahim Kamel
(Eds.)

Sponsored By:

- Purdue University
- University of California, Riverside
- University of Maryland Institute for Advanced Computer Studies
- Panasonic Technologies/PINTL
- Telcordia Technologies
Preface

This publication constitutes the proceedings of the Sixth International Workshop on Multimedia information Systems (MIS'00) held in Chicago, Illinois, October 26-28, 2000. This workshop continues a successful series of workshops that started in 1995 with the aim of fostering interdisciplinary discussions and research in all aspects of multimedia information systems research. Previous workshops were held in Arlington (VA), West Point (NY), Como (Italy), Istanbul (Turkey) and Palm Springs Desert (CA). As in the past, this workshop consists of participants drawn from a wide variety of disciplines such as: database systems, networking, operating systems, graphics and visualization, real-time systems etc., all of whom are engaged in research on one or more aspects of multimedia systems.

The workshop program includes three invited talks, two panels and 19 technical papers. Papers were selected from a collection of around 40 submitted papers through a peer review process. Each submitted paper was reviewed by four program committee members. The accepted papers are organized into five sessions namely:

- Video Storage and Retrieval
- Searching World Wide Web
- MM Networking and Streaming
- Multimedia Modeling
- Document and Image Retrieval

The invited talks are given by three experts in the area of multimedia systems. Don Towsley's (University of Massachusetts) talk is on “QoS, Proxies, and the Internet”, H.V. Jagadish (Univ. of Michigan, Ann Arbor) discusses “The Impact of XML on Multimedia”, and Alexander Hauptmann (Carnegie Mellon University) talks about “Multimedia Information Retrieval from a Digital Video Library”. The first panel discusses “Research Challenges in Multimedia Information Systems”, while the second panel discusses “From Minitel to the World-Wide Web and Beyond: The Ongoing Role of Multimedia Information Systems in Digital Government.”

The excellent collection of technical papers as well as the exciting invited talks and panels addresses a wide variety of timely issues on multimedia. We hope you will find the technical program stimulating, the discussions fruitful, and your stay in Chicago relaxing.

We would like to express our appreciation to all authors of submitted papers, to the MIS'00 Program Committee members, and to our invited speakers and panel participants. We thank Purdue University, the University of California, Riverside, University of Maryland Institute for Advanced Computer Studies, Panasonic Information and Network Technologies Laboratory and Telcordia Technologies for their support.

October 2000

Walid G. Aref
Purdue University

Ibrahim Kamel
Panasonic Technologies, Inc./PINTL
MIS-2000

(US Army Research Office, USA)
(University of Maryland, College Park, USA)
(University of California, Riverside, USA)

(University of Maryland, College Park, USA)

(Purdue University, USA)
(Panasonic Info. and Networking Technologies Lab., PINTL, USA)

(University of California, Santa Barbara, USA)
(Rensselaer Polytechnic Institute, USA)
(Georgia Tech, USA)
(Universita' di Milano, Italy)
(IBM T.J. Watson Research Center, USA)
(Purdue University, USA)
(Carnegie Mellon University, USA)
(Panasonic Info. and Networking Technologies Lab. PINTL, USA)
(Purdue University, USA)
(University of Southern California, USA)
(University of Delaware, USA)
(University of Ulm, Germany)
(University of California, Irvine, USA)
(Philips Research Labs, USA)
(University of California, Los Angeles, USA)
(New Jersey Institute of Technology, USA)
(Case Western Reserve University, USA)
(Bell Labs, USA)
(National Institute of Informatics, Japan)
(KAIST, Korea)
(University of California, Riverside, USA)
(ETH Zurich, Switzerland)
(George Mason University, USA)
(Princeton University, USA)
(Athens University, Greece)
(IBM T.J. Watson Research, USA)
(University of Alberta, Canada)
# Table of Contents

## Invited Talks

QoS, Proxies, and the Internet ................................................................. 1  
*Don Towsley*

The Impact of XML on Multimedia ......................................................... 2  
*H.V. Jagadish*

Multimedia Information Retrieval from a Digital Video Library ................... 3  
*Alexander Hauptmann*

## Video Storage and Retrieval

Efficient Multi-Resolution Video Partitioning in Video Databases .................. 4  
*H. Xu, E. El-Kwae, A. Younis, M. Kabuka*

Efficient Storage and Retrieval of News Video Data for NoD Services ............... 14  
*K. Jin and J. Chang*

A Two-Level Representation Model for Effective Video Data Storage ............... 23  
*A. Vakali and E. Terzi*

Motion-based Indexing and retrieval of Video Using Object Trajectories .......... 33  
*S. Dagtas, A. Ghafoor, and R. Kashyap*

## Searching World Wide Web

Avatar Programming in a Virtual Space on the World Wide Web ...................... 42  
*Y. Loh, T. Ogawa, M. Tsukamoto, and S. Nishio*

Web Search with Metadata Links and Multimedia Presentations ...................... 52  
*G. Ozsoyoglu, M. Anderson, and Z. M. Ozsoyoglu*

Query Rewriting for Multimedia XML Data ............................................. 62  
*J. Yoon, A. Hafez, and V. Raghavan*

WebMARS: A Multimedia Search Engine for Full Document Retrieval ................. 72  
*M. Ortega-Binderberger, S. Mehrotra, K. Chakrabarti, and K. Porkaew*

## MM Networking and Streaming

Network Level Framing in INSTANCE .................................................... 82  
*P. Halvorsen, T. Plagemann, and V. Goebel*

Video Multicasting with Variable Bandwidth Requirements .......................... 92  
*M. Samidi M. Faloutsos, R. Pankaj, and K. Sevcik*
MPEG-L/MRP: Adaptive Streaming of MPEG Videos for Interactive Internet
S. Boll, C. Heinlein, W. Klas, and J. Wandel

Managing Multimedia Streams in Distributed Environments using CORBA
V. Kalogeraki and D. Gunopulos

Multimedia Modeling

An Event Model for Video Interaction
C. Petermann and B. Wang

Modeling and Querying of Semantic - Content-Based Multimedia Databases
M. Abougalal, A. Saad, M. Ahmed, and M. Moharrum

Two Optimization Techniques to Improve the Performance of MBC-based Shape
M. Safar and C. Shahabi

Document and Image Retrieval

Geometric Shape Representation for Image Retrieval
M. Hacid, A. Rezgui, and J. Fan

A Technique for Self-embedding of Digital Images
V. Gerodetski, L. Popyack, and B. Skormin

Similarity Image Retrieval with Significance-Sensitive Nearest-Neighbor Search
N. Katayama and S. Satoh

Dynamic Graph Structured Document Databases
S. Adali, J. Liepins, and M. Sapino

Panel Sessions

Research Challenges in Multimedia Information Systems
Moderator: Sibel Adali

The Ongoing Role of Multimedia Information Systems in Digital Government: From Videotext to the World-Wide Web and Beyond
Moderator: William McIver

Author Index
Query Rewriting for Multimedia XML Data
Jong P. Yoon, Alaeldin Hafez, and Vijay Raghavan
The Center for Advanced Computer Studies
University of Louisiana, Lafayette, LA 70504-4330

ABSTRACT
Extensible Markup Language (XML) is emerging as a standard for representing and exchanging data in a variety of applications, each with its own special needs. It is, therefore, natural to explore the use of XML to represent multimedia data. While it is not difficult to customize XML for multimedia data, effective retrieval of information from a collection of multimedia document collection is not straightforward and a search may often result in either too many or too few hits. To cope with this problem, we propose a framework that enables retrieval from a multimedia document collection to be performed by rewriting user-given queries. We show how user-given queries, specified using typical querying interfaces, and several types of domain-specific rules may be represented in an XML tag-embedded query language. Using a document type definition (DTD) suitable for multimedia data and domain-specific rules, strategies for relaxing queries, based on input from the user with respect to the size and/or quality of result sets deemed acceptable, are provided. Finally, we propose suitable measures, termed accuracy and coverage, in order to evaluate the quality of rewritten queries. The contributions of this paper include mechanisms for (1) XML representation of multimedia queries and rules, (2) multimedia query rewriting, and (3) evaluation of rewritten queries.

Keywords: Adaptive query rewriting, Accuracy and Coverage of rewritten queries, Multimedia XML, Approximate information retrieval.

1. INTRODUCTION
Recently, eXtensible Markup Language (XML) has become an emerging standard for representing and exchanging multiple types of data. In XML, the document type definition (DTD) information is stored with the data. XML allows user-defined elements, nested elements, and an optional validation of document structure with respect to a DTD. Element names are called tags, and elements may also have attributes whose values are always atomic. In this paper, we focus on XML’s application to multimedia data. Multimedia data can be easily represented in XML. Multimedia XML Data (MXD) is therefore self-describing data. However, multimedia data retrieval is not straightforward and a query may obtain either too many, or too few hits occasionally.

EXAMPLE 1.1: Consider the following MXD example.
<doc>
<title>Blue Sky</title>
<author>James Smith</author> <author>John Doe</author>
<image>Lab</image>
</doc>

The root element multimedia document above contains the elements of title, authors, and the image of the lab. The "Lab" image element in turn contains the subcomponent image elements of "Keyboard," "Monitor," "Body," and "Mouse." We call this "container-only element." This container-only element is common in (especially multimedia) document collections. It may cause some difficulty in retrieving information efficiently and effectively. Suppose that a query is posed to display computers with mice where they are in the lab. The multimedia document as stated above is not matched with the query because the element image has no value of "Computer" exists. However, if subcomponent images of computers are used for query evaluation, the given multimedia document can be one of the matches. If adaptively matched, subcomponent images should be retrieved and composed to display for a requested image. To match with such container-only elements, this paper proposes a method of query rewriting.

Query rewriting has been developed for conventional databases. This earlier work has the following limitations in employing it for multimedia applications.
• Difficulty in dealing with multiple features retained in multimedia data. Multimedia data consists of multiple types of data and is tagged by nested elements. Typical approaches are not suitable for dealing with documents tagged by nested elements (or especially container-only elements).
Difficulty in making use of DTD (or schema information). Typical approaches are not able to deal with the relationship between component and subcomponent elements and their priorities.

This paper proposes a method of rewriting user-given queries to be processed against multimedia data. We consider the three possible types of user queries: (1) SQL-like queries, (2) Tag-embedded queries, and (3) Sketch-pad drawing queries. We believe that those types of queries are representative of known methods for querying a multimedia data. For a given multimedia query, the approach we propose in this paper (1) generates tag-embedded query, (2) poses it to a database and returns the matches, (3) relaxes the tag-embedded query with respect to user response (or feedback), and then (4) poses a revised query to the database. The steps (3) and (4) are repeated until users are satisfied.

The goals of this approach include:

- Approximate information retrieval - Multimedia data collections are usually very large and un-uniformly structured, which means that users cannot easily formulate pinpointed queries. Nevertheless, a result set of reasonable size is expected. The method proposed in this paper allows rewriting of user-defined queries for approximate retrieval.
- Adaptive information retrieval - The XML elements are prioritized, from popularly used elements to less popularly used ones. Such priority information is maintained for multimedia data collections. The XML elements in user-defined queries can be chosen based upon the priority information.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 describes preliminaries needed to describe the techniques proposed. Section 4 introduces three possible types of queries for multimedia data. Section 5 describes our method of query rewriting. Section 6 describes the measurements for rewritten queries. Section 7 concludes our work with future directions.

2. RELATED WORK

Literatures on query rewriting fall into three general approaches. (1) The first approach is syntactic query rewriting [BDHS96, Mot90]. User-given queries can be rewritten using structure hierarchy information if syntactically exact matching is not only one available in a database schema. (2) The second approach is semantic query rewriting [PV99, YK93]. User-given queries can be rewritten using semantic information like view, integrity constraints, and rules. The above two approaches are handled internally by a system. In contrast, (3) the third approach is user-interactive query rewriting [YK98]. User-given queries are rewritten directly by users or at least by using user inputs. This interaction continues until user-given queries are satisfied.

In general, unless stored in a typical database management system, multimedia data has no schema defined in advance to follow or no integrity constraints to be enforced [MAG+97]. Instead, Multimedia XML data follows DTD not in the sense of a database schema but in the sense of a type description. We exploit the syntax of a DTD for Multimedia XML data to rewrite queries. In addition, we use some heuristics to rank and choose from alternative ways of rewriting a given query. According to the user selection of the rank or the selection of rules about multimedia features, a query can be effectively rewritten. Our approach is in a way a combination of all approaches stated above.

3. PRELIMINARIES

In this section, we describe the basic concept of XML with DTD, rules in XML, and provide some definitions. A

```
1 <ELEMENT mndoc {life, author+, body, date}>  
2 <ELEMENT author {#PCDATA, {lastname,firstname} | fullname}>  
3 <ELEMENT body {para, image}>>  
4 <ELEMENT image {caption?, annotation*, pfeatures, ifeature, sfeature, image*, img}>>  
5 <ATTLIST image id CDATA #REQUIRED>  
6 <ELEMENT pfeature {#PCDATA}>  
7 <ATTLIST pfeature majorcolor CDATA #IMPLIED>
  bgcolor CDATA #IMPLIED
  height CDATA #IMPLIED
  width CDATA #IMPLIED
  size CDATA #IMPLIED>  
8 <ELEMENT feature {#PCDATA}>  
9 <ATTLIST feature position-grd CDATA #IMPLIED>
  abs-position {south|north|east|west} #IMPLIED
  rel-position {left|right|up|down|farleft|farright} #IMPLIED
  rel-position-with CDATA #IMPLIED
  overlapped {none|partially|totally} none
  overlapped-with CDATA #IMPLIED
  overlap-pivot {center|upperleft|lowerleft|upperright|lowerright} #IMPLIED>  
10 <ELEMENT sfeature {#PCDATA}>  
11 <ATTLIST sfeature texture CDATA #IMPLIED>
  tone CDATA #IMPLIED>
12 <ELEMENT img {#PCDATA}>  
13 <ATTLIST img src CDATA #REQUIRED>
  height CDATA #IMPLIED
  width CDATA #IMPLIED>  
```

Figure 3.1: DTD for MXD
part of the DTD example for multimedia data is in Figure 3.1. In DTD, there are five meta-characters, * for zero or more, ? for zero or one, + for one or more, | for one or the other, and comma (,) for consecutive order. Using those meta-characters, an element can have a number of component elements. A value of an element can be also a combination of the values of those component elements. An example of domain-dependent rules is that the term "computer" consists of "Keyboard," "Monitor," "Body," and "Mouse." Those components are also the terms defined as nested elements. Domain-dependent rules specify term-subterm relationships. Elements can be introduced to specify features associated with document components. For example, in Figure 3.1, we have elements corresponding to physical features (line 7), logical features (line 9), and semantic features (line 11). Now domain-dependent rules and feature-dependent rules can be defined in XML. A feature-dependent rule is defined as the physical feature pfeature majorcolor="#FFFFFF", meaning that the major color is white. Feature-dependent rules specify term-attribute relationships. An example is shown in Figure 3.2:

Such a rule is obtained by mining from XML instances, directly from users, or thesaurus.

**Definition 3.1: Domain-dependent Rule**: Domain-dependent rules are defined in an XML format, where for each element having the name "term", there is a value attached to that element. For any "term" element, the value of that element is implied by the values of its nested "term" sub-elements.

**EXAMPLE 3.1**: The interpretation of the domain-dependent rule in XML, say <term> A <term> B <term> C</term>, is B & C =⇒ A.

**Definition 3.2: Feature-dependent Rule**: Feature-dependent rules are defined in an XML format, where for each element having the name "term", there is a value attached to that element. For any "term" element, the value of that element is implied by the values of its nested "feature" sub-elements.

**EXAMPLE 3.2**: The interpretation of the feature-dependent rule in XML, say <term> A <pfeature majorcolor="#FFFFFF"/></term>, is pfeature.majorcolor="#FFFFFF" =⇒ A.

**Definition 3.3: Element Priority**: Priority of an element e, P = P(e), is a measure of the "importance" of e. Elements are derived by using information from DTD, actual XML document repository, or a mixture between DTD and actual XML document repository.

**EXAMPLE 3.3**: Assume the actual XML document repository X exists, and an element e=</doc><image> is one of the elements defined in X. Let N be the total number of documents in X, and n be the number of documents in X having e as an element with a value. The priority of element e can be defined as n/N.

**Expanded element**: An expanded element E is defined as a set of pairs \( \{(e_0, c_0), (e_1, c_1), \ldots, (e_{n-1}, c_{n-1})\} \), such that for any pair \( (e_i, c_i) \in E, 1 \leq i \leq n-1 \) there exists a pair \( (e_o, c_o) \in E, 1 \leq o \leq i-1 \), such that \( e_i \) and \( e_o \), and \( c_i \) and \( c_o \) are related either by structure topology, domain-dependent rules, or feature-dependent rules; (for more details, please see section 5.1.)

**Definition 3.4: Expanded element Priority**: The priority of an expanded element \( E = \{(e_0, c_0), (e_1, c_1), \ldots, (e_{n-1}, c_{n-1})\} \), is calculated by using the priority values of \( e_i, 1 \leq i \leq n-1 \), along with dependency information between \( e_i \)'s.

**Next element**: Next element to be added to an expanded element \( E = \{(e_0, c_0), (e_1, c_1), \ldots, (e_{n-1}, c_{n-1})\} \), is defined as
next\text{-}element (E) = \begin{cases} 
\emptyset & \text{if } \forall \text{elements } e \in E \text{ there is no more related elements} \\
(e_n, c_n) & \text{if } \exists n \text{ such that } e \text{ is related to an element } e_i \in E, \\
\text{value of } c \text{ is determined from the method used} & 
\end{cases}

4. MULTIMEDIA SEARCHES AND QUERIES

Possible mechanisms to display "computers" from multimedia data include: (1) by text search to find documents containing "computers," (2) by semantic features, i.e., feature in the DTD, or the annotation literally stating "computers" to find computer images in multimedia documents, (3) by image recognition to find the image of the letter "computer," (4) by the physical features feature about color, say a square-shaped white-object, to find computer images, (5) by logical features feature about relationships among component objects, say a monitor, a key-board, a mouse, and a unit, (6) by looking external documents, the documents linked by the element img in the DTD.

To handle these possible searches, we consider the following three queries: (1) SQL-like queries [AQM+96], (2) tag-embedded queries [DFLS98], (3) Sketch-pad drawing queries [WLC+99]. These queries can be defined as follows:

**Definition 4.1:** For a query \( Q \) in the form \( Q = (\forall_1 O_1 \ C_1 Y_1 (V_1, O_1 \ C_1 Y_1) \ldots (V_i, O_i \ C_i Y_i) / \ldots / V_m, O_m \ C_m Y_m) \)

where \( 0 \leq i \leq m-1 \), \( V_i \) is element name, \( O_i \) is a comparison operator, \( C_i \) is a value, and \( l_i \) is a logical operator.

**Example 4.1:** Consider the query example in EXAMPLE 1.1 in the three query languages. An SQL-like query specifies \( \text{SELECT } m \text{ FROM doc.image WHERE m.image } = \text{ 'Computer' AND m.image } = \text{ 'Mouse'; then the equivalent query } Q \text{ is written as } Q = (\{\text{doc}<\text{image} = \text{ 'computer'}\} \land (\text{doc}<\text{image} = \text{ 'Mouse'}\}) \).

A tag-embedded query specifies \( \text{WHERE } <\text{doc}<\text{image} = \text{ 'Computer'}\land <\text{image} = \text{ 'Mouse'}\land <\text{image} = \text{ 'Computer'}\land <\text{image} = \text{ 'Mouse'}\land <\text{image} = \text{ 'Computer'}\land <\text{image} = \text{ 'Mouse'}\) on the other hand, one example of sketch-pad drawing queries is in Simplicity project (http://ixw.stanford.edu/cgi-bin/zwang) as shown in Figure 4.1. That query can be converted into a tag-embedded query: \( \text{WHERE } <\text{image} = \text{ 'Mouse'}\land <\text{shape} = \text{ 'Rectangle'}\land <\text{color} = \text{ 'White'}\land <\text{color} = \text{ 'Black'}\land <\text{border} = \text{ 'Red'}\land <\text{border} = \text{ 'Yellow'}\land <\text{fill} = \text{ 'Blue'}\land <\text{fill} = \text{ 'White'}\land <\text{fill} = \text{ 'Red'}\land <\text{fill} = \text{ 'Yellow'}\) in this case, \( Q \) is \( (<\text{image} = \text{ 'Mouse'}\land <\text{shape} = \text{ 'Rectangle'}, <\text{color} = \text{ 'White'}\land <\text{color} = \text{ 'Red'}\land <\text{color} = \text{ 'Blue'}\land <\text{color} = \text{ 'Yellow'}\land <\text{shape} = \text{ 'Rectangle'}, <\text{shape} = \text{ 'Round'}\land <\text{shape} = \text{ 'Square'}\land <\text{shape} = \text{ 'Triangle'}\land <\text{shape} = \text{ 'Quadrilateral'}\)\). 

![Figure 4.1: Example of Sketch-pad Drawing Queries](image)

For a user-defined query, documents are matched and retrieved. The answer to a query is returned to users. The user reaction to an answer can be one of the following: (1) Simple acceptance by a user. In this case, a system does not need to query rewriting. (2) System-driven query rewriting. In this case, a system can do for query relaxation or restriction with minimal user input. (3) Query rewriting by relevance feedback. A user chooses one of the possible choices and the system performs query relaxation or query restriction, according to user input. These queries can then be rewritten as described in the following section.
5. QUERY REWRITING

5.1 Intuition

Referring to Section 3, a part of the DTD (lines 1, 2, 3, and 4) may be simplified to

\begin{align*}
1 & \quad m = ta + bd7 \\
2 & \quad a = (0)u \\
3 & \quad b = (h)u \\
4 & \quad i = e^n \ast p^x \ast t^y \ast f^z \\
\end{align*}

Consider a query condition, specified by query elements: `<image><image> A </image><image>`, `<para>B</para>`, which simply says, ii = “A” & h = “B”. In adaptive information retrieval, users may have three possible reactions:

- Simple acceptance by a user. Whatever results matched with a query are returned to a user.
- System-driven query rewriting. A user asks the system to re-evaluate a query if the query is yet to be relaxed. In this case, documents may have the query element in different level or they may have no query element but its component elements. To retrieve the former documents, we want to relax this query element to include more levels of elements. By using elements priorities, i.e., starting with query elements having highest priorities, three options are considered,
  - Structure topology, a query element may include higher or lower level of elements, e.g., Q: (i = “A” OR ii = “A” OR iii = “A”) & h = “B”.
  - Domain-dependent rules, a query element may include component elements, e.g., the computer image ii = “A” & h = “B”, and the component images of Keyboard, Monitor, Body, and Mouse are displayed, Q: (i = “A” OR (ii = “A1” & ii = “A2” & iii = “A3” & iii = “A4”)) & h = “B”, where {A1, A2, A3, A4} = {A}. In this case, all component images need to be composed to present a complete image [BFS00].
  - Feature-dependent rules, a query element may be replaced by feature elements, e.g., the computer image ii = “A” & h = “B”, and the images with ip.c = “W” are displayed, Q: (ii = “A” OR (ip.c = “W”)) & h = “B”.

After relaxing a query element q, a new priority value should be recalculated for element q. For any of the above three options, another phase of relaxation could be carried by replacing “&” operations with “OR” operations. This phase will be discussed in section 5.3.

- Interactive query rewriting. This approach is similar to system-driven query rewriting. The system may provide users with an ordered list of queries based on priority of query elements along with either the structure expansion option, the domain-dependent rules option, or the feature-dependent rules option. Suppose the element <image><image><image><image> $m$ </image><image><image><image> has higher priority than element <image><image><image><image> $m$ </image><image><image><image>, then our system may provide the user with the priority of query elements in an order, say <image><image><image><image> $m$ </image><image><image><image>, <image><image><image><image> $m$ </image><image><image><image>, <image><image><image><image> $m$ </image><image><image><image>, <image><image><image><image> $m$ </image><image><image><image>, from the highest priority to the lowest priority. The user may choose one of the options in the rank, say <image><image><image><image> $m$ </image><image><image><image>, and ask the system to re-evaluate the query.

5.2 Query Condition Rewriting

For a query $Q = (V_o, O_o, C_o) \ldots (V_i, O_i, C_i) \ldots \ldots (V_{m-o}, O_{m-o}, C_{m-o})$, the method we discuss is to rewrite $v$ and $c$. Among the three options, $v$ can be rewritten by the first option, while $c$ is rewritten by the third option: (1) Using structure topology, (2) domain-dependent rules, (3) feature-dependent rules. Each step of these is pictorially depicted in Figure 5.1.

![Figure 5.1: Query Rewriting](image)

Notice that in Figure 5.1, any combination of these three options are taken by either the two of three user reactions: system-driven query rewriting or adaptive query rewriting, which is described in Section 5.1.

66
5.2.1 Condition Rewriting using Structure Topology
A DTD represents element hierarchies. The structure topology of element hierarchies of a DTD can be used to relax user-given queries. This option is illustrated in Figure 5.1(1). Given DTD, C=i*b, i=i*, and a query Q: i=i="A", the query Q can be rewritten Q: i="A' OR i=i="A' OR i=i="A'"
For the meta-characters, *, ?, +, |, we can consider the four primitive query elements: {C1=i+b, i=i*}, {C2=i?b, i=i*}, {C3=i+i, i=i*}, and {C4=i+b, i=i*}. The illustration of some examples of query relaxation using structure topology is shown in Table 5.1. Notice that all characters represent simplified elements.

<table>
<thead>
<tr>
<th>DTD</th>
<th>Structure topology</th>
<th>User-given query</th>
<th>Relaxed query example</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1=i+b</td>
<td>(b, i<em>b, i</em>b)</td>
<td>i = &quot;A' &amp; b = &quot;B&quot;</td>
<td>(i = &quot;A' OR i = &quot;A&quot;) &amp; b = &quot;B&quot;</td>
</tr>
<tr>
<td>C2=i?b,</td>
<td>(i,ib,ibc)</td>
<td>i = &quot;A&quot; &amp; b = &quot;B&quot;</td>
<td>b= &quot;B&quot;</td>
</tr>
<tr>
<td>i=i*</td>
<td></td>
<td>i = &quot;A&quot;</td>
<td></td>
</tr>
<tr>
<td>C3=i+i,</td>
<td>(ib, ib, ibb)</td>
<td>i = &quot;A&quot; &amp; ib = &quot;B&quot;</td>
<td>(i = &quot;A&quot; OR i = &quot;A&quot;) &amp; ib = &quot;B&quot;</td>
</tr>
<tr>
<td>i=i*</td>
<td></td>
<td>i = &quot;A&quot; OR i = &quot;B&quot; &amp; ib = &quot;B&quot;</td>
<td></td>
</tr>
<tr>
<td>C4=i+b,</td>
<td>(i, b, bc)</td>
<td>i = &quot;A&quot; &amp; b = &quot;B&quot;</td>
<td>i = &quot;A&quot; OR b = &quot;B&quot;</td>
</tr>
<tr>
<td>i=i*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: Query Rewriting using Structure Topology

5.2.2 Condition Rewriting using Domain-dependent Rules
Another option in Figure 5.1(2) is to use domain-dependent rules to rewrite user-given queries. As defined in Section 3, domain-dependent rules specify the a-part-of relationship between a term and a number of component terms. For example, assume that DTD, C=i*b, i=i*, and a domain-dependent rule, <term> A <term> B <term> C <term> D <term>. A given query Q: i=i="A" can be rewritten Q: i= "A' OR (i=i="B" & i=i="C") & i=i="D") & b=i=""B" Some examples of query relaxation using domain-dependent rules are illustrated in Table 5.2.

<table>
<thead>
<tr>
<th>DTD</th>
<th>Domain-dependent rules</th>
<th>User-given query</th>
<th>Relaxed query example</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1=i*b</td>
<td>&lt;term&gt; A</td>
<td>i = &quot;A' &amp; b = &quot;B&quot;</td>
<td>(i = &quot;A' OR (i = &quot;C&quot; &amp; i = &quot;D&quot;) &amp; b = &quot;B&quot;</td>
</tr>
<tr>
<td>i=i*</td>
<td>&lt;term&gt; C &lt;term&gt; D &lt;term&gt;</td>
<td>i = &quot;A' &amp; b = &quot;B&quot;</td>
<td>(i = &quot;A' OR (i = &quot;C&quot; &amp; i = &quot;D&quot;) &amp; b = &quot;B&quot;</td>
</tr>
<tr>
<td>C3=i+b,</td>
<td></td>
<td>i = &quot;A' &amp; ib = &quot;B&quot;</td>
<td>(i = &quot;A' OR (i = &quot;C&quot; &amp; i = &quot;D&quot;) &amp; ib = &quot;B&quot;</td>
</tr>
<tr>
<td>i=i*</td>
<td></td>
<td>i = &quot;A' OR (i = &quot;C&quot; &amp; i = &quot;D&quot;) &amp; ib = &quot;B&quot;</td>
<td></td>
</tr>
<tr>
<td>C4=i+b,</td>
<td></td>
<td>i = &quot;A' &amp; b = &quot;B&quot;</td>
<td>i = &quot;A' OR b = &quot;B&quot;</td>
</tr>
<tr>
<td>i=i*</td>
<td></td>
<td>i = &quot;A' OR (i = &quot;C&quot; &amp; i = &quot;D&quot;) &amp; b = &quot;B&quot;</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Query Rewriting using Domain-dependent Rules

5.2.3 Condition Rewriting using Feature-dependent Rules
The third option is to use feature-dependent rules for query relaxation as shown in Figure 5.1(3). As defined in Section 3, feature-dependent rules specify the physical, logical, or semantic features about an element (or a term).

<table>
<thead>
<tr>
<th>DTD</th>
<th>Feature-dependent rules</th>
<th>User-given query</th>
<th>Relaxed query example</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1=i*b</td>
<td>&lt;term&gt; A</td>
<td>i = &quot;A' &amp; b = &quot;B&quot;</td>
<td>(i = &quot;A' OR i = &quot;Z&quot;) &amp; b = &quot;B&quot;</td>
</tr>
<tr>
<td>i=i*</td>
<td></td>
<td>i = &quot;A' OR i = &quot;Z&quot; &amp; b = &quot;B&quot;</td>
<td></td>
</tr>
<tr>
<td>C2=i*b,</td>
<td>&lt;pf feature z=&quot;Z&quot; /&gt;</td>
<td>i = &quot;A' &amp; b = &quot;B&quot;</td>
<td>(i = &quot;A' OR i = &quot;Z&quot;) &amp; b = &quot;B&quot;</td>
</tr>
<tr>
<td>i=i*</td>
<td></td>
<td>i = &quot;A' OR i = &quot;Z&quot; &amp; b = &quot;B&quot;</td>
<td></td>
</tr>
<tr>
<td>C3=i+b,</td>
<td></td>
<td>i = &quot;A' &amp; ib = &quot;B&quot;</td>
<td>(i = &quot;A' OR i = &quot;Z&quot;) &amp; ib = &quot;B&quot;</td>
</tr>
<tr>
<td>i=i*</td>
<td></td>
<td>i = &quot;A' OR i = &quot;Z&quot; &amp; ib = &quot;B&quot;</td>
<td></td>
</tr>
<tr>
<td>C4=i+b,</td>
<td></td>
<td>i = &quot;A' &amp; b = &quot;B&quot;</td>
<td>i = &quot;A' OR b = &quot;B&quot;</td>
</tr>
<tr>
<td>i=i*</td>
<td></td>
<td>i = &quot;A' OR b = &quot;B&quot;</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3: Query Rewriting using Feature-dependent Rules
Such features are useful for content-based information retrieval. For example, assume that DTD, C=i*b, i=i*, and a feature-dependent rule, <term> A <term> B <term> C <term> <p feature.majorcolor="#FFFFFF" /> <term>. A given query Q: i="A" can be rewritten Q: i="A" OR (i.p.feature.majorcolor="#FFFFFF"). Some examples of query relaxation using feature-dependent rules are illustrated in Table 5.3.

A user-given query, if not satisfied, can be rewritten by any combination of the above three options. Once it is rewritten, the rewritten query is processed against multimedia XML data as shown in Figure 5.2.

![Figure 5.2: Rewritten Query Answering](image)

### 5.3 Algorithm

This section describes an algorithm for query relaxation and discusses its complexity. The relaxation procedure has two phases:

- The expansion phase, and
- The inclusion phase.

**Algorithm Relax** *(Q)*

**Initialization:**

- For a query \( Q=(V_0, O_0, C_0), (V_1, O_1, C_1), \ldots, (V_m, O_m, C_m) \), \( E_i = \{ (V_i, C_i) \} \), \( \text{Exp} = \text{true}, \text{incl} = \text{true} \), and \( U = \cup_i (E_i, P_i) \), 0 \( \leq i \leq m-1 \).

**begin**

1: choose \((E_i, P_i)\) ∈ \( U \) such that \( P_i \) is the largest and \( (\text{Exp} = \text{true} \) or \( \text{incl} = \text{true} \))

   // query element \( E_i \) is having highest priority and can be expanded or operator \( l \) is "AND"

2: while \( \text{Exp} \)

   **begin**

   Expansion\((Q, E_i)\);
   if \( \text{Exp} \) Execute \( Q \); quit;
   end;

   Inclusion\((Q, E_i)\);
   if \( \neg \text{incl} \) end; // no more relaxation for \( E_i \);
   Execute \( Q \);
   quit;

   **end.**

**Expansion** *(Q, E_i)*

**Begin**

choose a method \( M \), such that next-element\(_\alpha(E_i)\neq\emptyset \), and

- the Accuracy of the new formed query is the highest;
- three methods are considered, Structure topology, Domain-dependent rules,
- and Feature-dependent rules.

\( E_i \leftarrow E_i \cup (\text{next-element}(E_i), C_i) \);

// \( C_i \) is a method dependent value, e.g., \( C_i=C_i \) in structure topology method;

Recalculate \( P_i \);

\( U \leftarrow U \cup ((E_i, P_i)) \);

\( E_i \) elements in \( Q \leftarrow E_i \) elements in \( Q \cup (\text{next-element}(E_i), O, C_d) \);

If for all methods \( M \), next-element\(_\alpha(E_i)\)= \( \emptyset \),

**Begin** \( \text{Exp} = \text{false}; \) quit; **end**;

**end;**

![Figure 5.3: Query Relaxation Algorithm](image)

68
**Figure 5.3 (cont.): Query Relaxation Algorithm**

As shown in figure 5.3, the procedure alternates between the expansion phase and the inclusion phase according to priority values and expansion fulfillment. In the expansion phase, the system initially creates an expanded element $E_i$, with length 1, for each element $F_i$ in $Q$, then a set $U$ of all pair tuples $(E_i, F_i)$ is initialized. The main idea behind the expansion phase is, query elements are expanded by adding a new element $e$ to the query $Q$ by either following the structure topology, i.e., $e = \text{next-element}(E_i)$, using the domain-dependent rules, or using the feature-dependent rules. The new element $e$ is an eligible element descendant of an expanded element $E_i$ in $Q$, and $E_i$ has the largest available priority. In the expanded query $Q$, those elements in $E_i$ are "ORed" with $(e, O_i, C_i)$. After generating a new relaxed query $Q$, either the expansion phase or the inclusion phase can be used, with the latest relaxed query to generate a new more relaxed query. For a query element $q$ with the highest priority, the expansion phase stops when there is no more expansion could be generated, e.g., all next-elements are null.

In the inclusion phase, we use the set theory's inclusion of sets. For two sets $A$ and $B$, $A \cup B$ is included in $A$ (or $B$), and $A$ (or $B$) is included in $A \cup B$. By using the priority values, we start with the "&" operation that having the highest priority (in the inclusion process, we only convert "AND" operations). Let $A \cap B$ be an "&" operation having the highest priority, and $P(A) > P(B)$. The new relaxed query $Q$ is generated by eliminating $B$ from the predicate $A \cap B$. For a next more relaxed query, $A \cap B$ should replace $A$ in $Q$.

Each time a new relaxed query is generated, the related coverage and accuracy values are calculated. In order to determine the significance of the new query, users should define the levels of coverage and accuracy needed. According to their input, the relaxation procedure is repeated to meet their significance input. Although we are referring only to query relaxation, the same concepts discussed above could be applied on query restriction.

In Algorithm Relax($Q$), we start with choosing the query predicate $E_i$ with the highest priority value, then the expansion module Expansion($Q$, $E_i$) is called to generate a prospective next-element for each of the three methods: Structure topology, Domain-dependent rules, and Feature-dependent rules. The Accuracy value is calculated for each of the expected methods, and the algorithm chooses the method with the highest Accuracy value. If a query predicate $E_i$ is chosen and there is no more expansions, the inclusion module Inclusion($Q$, $E_i$) is called to choose a predicate $E_i$, and apply the inclusion function.

**EXAMPLE 5.1:** For the query $Q = (\langle doc \rangle \langle image \rangle = 'computer' )$ and $(\langle doc \rangle \langle title \rangle = 'Blue Sky')$, let the DTD structure and the rule definition in Figures 3.1 and 3.2, respectively, be used for the methods: Structure topology, Domain-dependent rules, and Feature-dependent rules. In Figure 5.4, we give a sample of queries that could be generated by the relaxation.

1. if Structure topology method has the highest accuracy
   $Q_1 = (\langle doc \rangle \langle image \rangle = 'computer' \ OR \ \langle doc \rangle \langle image \rangle = 'computer' ) \ & \ (\langle doc \rangle \langle title \rangle = 'Blue Sky')$,

2. if Domain-dependent rules method has the highest accuracy
   $Q_2 = (\langle doc \rangle \langle image \rangle = 'computer' \ OR \ \langle doc \rangle \langle image \rangle = 'Keyboard' \ & \ \langle doc \rangle \langle image \rangle = 'Monitor' \ & \ \langle doc \rangle \langle image \rangle = 'Body' \ & \ \langle doc \rangle \langle image \rangle = 'Mouse' ) \ OR \ \langle doc \rangle \langle title \rangle = 'Blue Sky')$.

**Figure 5.4: Relaxed Queries Sample.**
6. MEASUREMENTS OF REWRITTEN QUERIES

By following algorithm Relax(Q), Q = (V_0, O_0, C_0, l_0), (V_1, O_1, C_1, l_1), ..., (V_m, O_m, C_m, l_m), the Relaxation procedure can generate at most \( m = \sum_{i=1}^{m} \text{maximum_extensions}(E_i) + m \) relaxed queries for Q, where maximum_extensions(E) is the maximum number of terms that could be added to E by all methods. The complexity of the Relaxation procedure is O(m).

We employ the notion of coverage and accuracy to compute the following measurements.

Two measures are proposed to define the significance of a generated query to the original one [DRS95, HS94]. These measures are

\[
\text{Coverage} = \frac{|Q_0 \land \sigma_D(X)|}{|Q_0(X)|}, \quad \text{and} \quad \text{Accuracy} = \frac{|Q_0 \land \sigma_D(X)|}{|\sigma_D(X)|},
\]

where \( \sigma_D(X) \) and \( \sigma_D(X) \) are the results generated from applying the original query \( Q \) and a relaxed query \( D \), and \( |Q_0(X)| \), \( |\sigma_D(X)| \), and \( |Q_0(X)| \land \sigma_D(X) \) are the sizes the results generated from applying the original query, a relaxed query, and the common results.

**Lemma 6.1:** The Relaxation procedure always generates relaxed queries with Coverage=1.

**Proof:** Following the Relaxation procedure, \( \sigma_D(X) \) is always included in \( \sigma_D(X) \), i.e.,

\[
|Q_0 \land \sigma_D(X)| \leq |Q_0(X)|. \quad \text{Then Coverage can be written as Coverage} = \frac{|Q_0 \land \sigma_D(X)|}{|Q_0(X)|}. \quad \text{and} \quad \text{Accuracy} = \frac{|Q_0 \land \sigma_D(X)|}{|\sigma_D(X)|} = 1. \quad \square
\]

**Lemma 6.2:** Let \( Q_0, Q_1, ..., Q_j \) be a set of relaxed queries generated by the Relaxation procedure for a query \( Q \). For any two relaxed queries \( Q_i \) and \( Q_j \), \( 1 \leq i, j \leq r-1 \), \( \text{Accuracy}(Q_j, Q_i) = \text{Accuracy}(Q_i, Q_j) \).

**Proof:** Following the Relaxation procedure, \( |Q_0(X)| \) is always included in \( \sigma_D(X) \), i.e.,

\[
|Q_0(X) \land \sigma_D(X)| = |Q_0(X)|. \quad \text{The Accuracy of query} \ Q_i \text{ can be written as Accuracy} = \frac{|Q_0(X) \land Q_i(X)|}{|Q_0(X)|}. \quad \text{Since query} \ Q_j \text{ is more relaxed than query} \ Q_i \text{ and the answer of} \ Q_i \text{ is included in the answer of} \ Q_j, \text{ then} \quad \frac{|Q_0(X)|}{|Q_0(X)|} > \frac{|Q_0(X)|}{|Q_0(X)|}. \quad \square
\]

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 \( \text{AvB} \) to \( A \) then to \( A \cdot B \). Also, in query restriction, lemmas 6.1 and 6.2 are switched in a sense that The Restriction procedure always generates restricted queries with Accuracy=1, and for any two restricted queries \( Q_i \) and \( Q_j \), \( 1 \leq i, j \leq r-1 \), \( \text{Coverage}(Q_i, Q_j) > \text{Coverage}(Q_j, Q_i) \).

Several techniques could be used to calculate 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.

---

70
• 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.

7. CONCLUSION

In the context of information retrieval from a collection of multimedia documents, the need for the use of query rewriting strategies is established, through motivating examples. Specifically, we assume that XML is extended to handle multimedia data and a collection of multimedia XML data (MXD) is available.

Query rewriting is proposed as a way of achieving approximate and adaptive information retrieval, since there is usually too much variability in the content of MXD to expect that user can specify their queries precisely. For enabling query rewriting, user given queries and several types of domain-specific rules are represented as tag-embedded XML documents. An algorithm for relaxing queries, which involves the use structure topology, domain- and feature-dependent rules and user input on the desirable quality of the rewritten query, is provided. Measures for evaluating the quality of rewritten queries are introduced. The algorithm can easily be generalized to handle certain kinds of query restriction.

ACKNOWLEDGEMENT

This research was in part supported by the U.S. Department of Energy grant DE-FG02-97ER1220 and La LEQSF research grant No LEQSF(2000-03)-RD-A-42. The first author is currently on leave of absence from Sookmyung W. University, Seoul, Korea.

REFERENCES


