A Constraint-based Object Model for Structured Document Management[0]

Anisoara Nica and Elke Angelika Rundensteiner
Dept. of Elect. Eng. and Computer Science; Software Systems Research Laboratory
University of Michigan, Ann Arbor, MI 48109-2122, 1301 Beal Avenue
e-mail: anica@eecs.umich.edu, rundenst@eecs.umich.edu

Abstract

Complex multimedia document handling, including the modeling, decomposition, and search across digital documents, is one of the primary services that must be provided by digital library systems. In this paper, we present a general approach for handling structured documents (e.g., SGML documents) by exploiting object-oriented database technology. For this purpose, we propose a constraint-based object model capable of capturing in a uniform manner all SGML constructs typically used to encode the structural organization of complex documents. We present a general strategy for mapping arbitrary document types (e.g., article, journal, and book DTDs) expressed using standard SGML into our model. Most importantly, we demonstrate that our model is designed to handle the integration of diverse document types into one integrated schema, thus avoiding the generating of numerous redundant class definitions for similar document subtypes. The resulting document management system DMS is thus capable of supporting the dynamic addition of new document types, and of uniformly processing queries spanning across multiple document types. In this paper, we also describe the implementation of our approach on the commercial DBMS system Illustra to demonstrate that the ease with which our approach can be realized on current OODB technology – without requiring any special-purpose constructs. Our DMS system provides support for integrated querying of both structural as well as content-based predicates across arbitrarily complex document types.

1 Introduction

1.1 Structured Document Management

With the coming of the information age, the amount, complexity and variety of digital documents available on the so-called information super-highway is increasing dramatically on a day by day basis. To cope with this overload of digital information, it is thus critical that we have at our avail a proper set of software tools to aid with all aspects of document handling. This includes tools and techniques for the creation, editing, management, storage, filtering, retrieval and presentation of structured multimedia documents.

One of the primary goals of the University of Michigan Digital Library (UMDL) project [1], funded by NSF, ARPA, NASA as well as numerous industrial sponsorers, is to develop an agent-based software

[0]This research has been funded in part by the joint NSF/ARPA/NASA Digital Libraries Initiative under CERA IRI-9411287, and by NSF under grants RIA #IRI-9309076 and NYI #IRI-9457609. We also thank Illustra Inc. to provide us with the University Innovation Equipment Award.

architecture for maintaining such digital information distributed over a large set of heterogeneous collection databases. The UMDL system is being designed to support the management and efficient retrieval of digital information over the internet. This requires techniques for identifying digital material that most closely matches the information needs of individual users – a functionality typically provided by information retrieval (IR) systems. However, documents of the current and future information age are no longer simple sequential pieces of text, mimicking their old cousin – the paper book. Rather, digital documents are complex, highly structured artifacts, that are closely interconnected referring and re-using passages and digital subdocuments in an integrated web structure. The challenge thus remains to not only identify documents as a whole, but rather to find individual document fragments, to search based on the structure of a document, and to compose document fragments into complex, customized documents targeted to meet the particular information needs of the users. DBMSs have been designed to handle such complex structured query and information integration requirements – besides of course also providing other traditional DBMS functionalities such as concurrency control and recovery.

For these reasons, much work has been initiated in recent years in integrating database (DBMS) and information retrieval (IR) technologies for developing structured document management systems [8, 9, 4, 3, 2, 7]. This research has generally proceeded in two directions: either IR systems are extended with structured representation and associated query techniques or DBMS systems are extended by incorporating IR matching techniques into their query language and access structures. It is likely that eventually a true 'marriage' will emerge once both approaches have fully matured. In this paper, we built on this previous work, in particular, we utilize a DBMS system that has been extended with IR techniques as foundation of our document management system (DMS) system.

Our goal is to provide for document management technologies using generic techniques and tools, whenever possible. For this reason, our DMS is designed to handle SGML documents. SGML (Standard Generalized Markup Language) [11] is a recently prevailing standard that has been developed as a means for the generic interchange of digital documents between different platforms and systems. SGML encodes different types of textual documents (e.g., document type definitions - DTDs - for articles, books, journals, etc.) by marking their structural contexts. It has been found that SGML structure can be exploited for guiding the retrieval process over digital documents, by allowing for example queries that incorporate not just content-based information needs but also structural requests. Examples of queries that our DMS can support, by exploiting SGML structuring, include *'Retrieve all figures with associated captions and the names of the sections they appear in for all documents written by Elisa Meister that are on the subject of "Database" and "Information Retrieval".'*

1.2 The Constraint-based Approach

This paper describes our solution to building a document management system (DMS) that can handle complex mixed queries by exploiting both IR as well as DBMS techniques. Our goals here are two-fold: (1) to build upon previous work whenever possible, and (2) to develop generic solutions that can be quickly adapted by others into their respective document management systems. Our work utilizes SGML as document structuring technique due to the increasing popularity of this standard among publishers[1]. In order to simplify mapping of SGML DTDs into database technology, we propose a constraint-based object modeling and merging approach.

First, we introduce a formal object model which (containing only well-accepted object modeling primitives) can easily be mapped to most of the currently available OODB systems[2]. Rather than introducing

[1] The digital library project at the University of Michigan (UMDL) is getting delivered document collections in SGML format from content providers such as Elsevier and others.

[2] We have previously identified constraint mechanisms as a powerful object modeling technique[14].

special-purpose data types and meta-classes or requiring new query processing techniques for handling particular SGML-constructs, as proposed by other researchers, we introduce a constraint-based object model that provides a generic mechanism for document modeling. We then present a general algorithm for mapping any SGML construct into our formal constraint-based object model [3]. Finally, we discuss our successful experience of implementing our DMS system on the commercial system Illustra [4]. This effort demonstrates that our formal constraint-based object model can easily be mapped to available OODB technology.

Furthermore, we show that our model is designed to support the integration of multiple DTDs into one integrated system – sharing class definitions among different DTDs whenever possible. Our integrated schema approach is based on the recognition that the textual document types that we are currently considering, such as, articles, books, journals and so on, typically exhibit significant overlap in terms of their document structures. We thus propose to exploit this fact by merging schemata rather than maintaining a distinct schema for each separate DTD in DMS – as commonly assumed by all the other systems that we are aware of. This not only reduces the size of the schema, but it can also simplify document retrieval by allowing for the structural and content-based queries against the integrated schema – i.e., across diverse DTDs. Specification of queries across various document types becomes possible without requiring special syntactic query constructs or query processing techniques.

To validate our constraint-based approach, we have implemented a working prototype of our document management system DMS. The foundation of our system is Illustra, a commercial object-relational DBMS, which we chose for its support of basic object-oriented features, of SQL and textual query processing, and of rule specification. Given this suitable platform, we were able to rapidly build an initial version of DMS within a few months. In this paper, we describe in detail a realization of the SGML-to-OODB mapping strategy on our chosen platform. Our prototype DMS can handle mixed queries like the one listed earlier. To summarize, key contributions of this proposed approach are its genericity and simplicity, which allow structured document handling to be easily added to any OODBs system that supports constraints.

1.3 Overview

In the remainder of this paper, we first overview our general approach towards document management (Section 2). Section 3 introduces the constraint-based object model, while Section 4 describes our proposed strategy for mapping SGML model groups into the constraint-based model. Each of these steps is clarified by presenting a detailed example of how it is realized in our current prototype implementation using Illustra. Our model simplifies the task of merging multiple, possibly diverse DTDs into one uniform schema, as described in Section 5. Finally, related work is presented in Section 6, and conclusions and future work are given in Section 7.

2 Project Background

Our goal is to develop an integrated document management system (DMS) that can handle mixed queries involving both structural as well as content-based predicates. The general architecture of our system is given in Figure 1.

Using SGML for Document Management. The SGML standard [10, 11], developed to support the representation and exchange of structured text, such as book or article document types, has increasingly

[3] It is important to note that most commercial DBMS systems provide some form of rule or constraint support, and that active OODBMS technology is increasingly gaining popularity [16, 12].

[4] Illustra is a registered trademark of Illustra Inc.

become popular with major publishers. We hence have chosen SGML as the formalism for encoding the structural characteristics of our documents. In other words, our DMS system is designed to accept SGML documents as input, to generate SGML documents as output, and to provide support for querying, editing, and filtering of documents based on their structural characteristics as encoded by SGML (Figure 1).

Mapping the structure of SGML documents into a database requires that the database model supports (1) the hierarchical structure of the SGML documents, (2) abstract data types such as variable-length ordered lists and tuples, (3) complex data types which can capture the semantics of the expressions obtained using SGML connectors ("|", "&", ",") and SGML occurrence indicators ("+", "?", "*") and (4) ways of defining constraints for the above data types [7]. In Sections 3 and 4, we present our constraint-based object modeling and merging approach which represents our solution to the problem of SGML modeling.

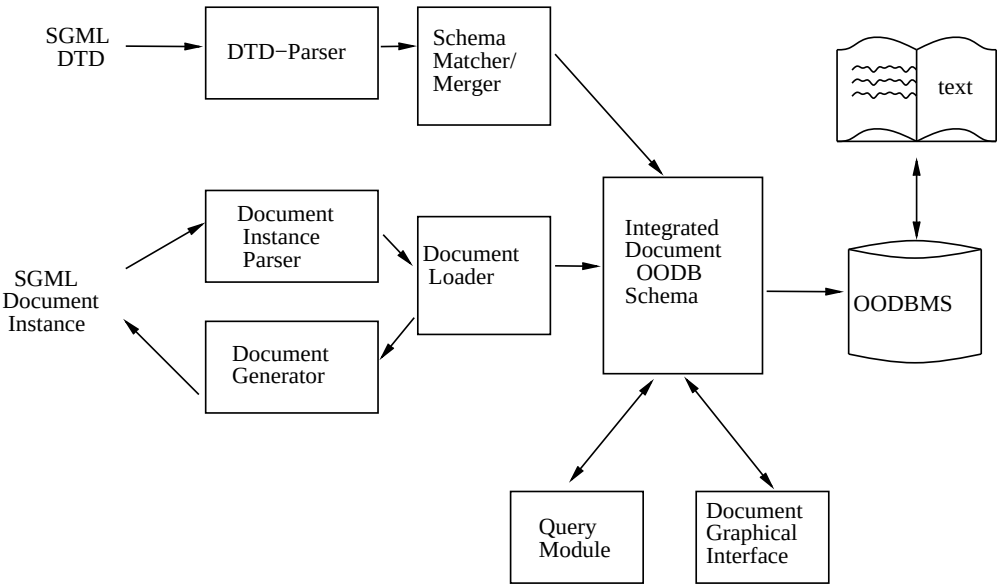

Figure 1: The Integrated Document Management System Architecture.

Selection of DBMS System as Platform. Since the complex structure of digital documents requires powerful data modeling, we have chosen the commercial DBMS system Illustra as platform for DMS. Illustra is based on an object-relational model which builds on top of the relational model through its support for objects and abstract data types. Thus, Illustra combines the mature features of relational databases (such as SQL query support) with the object modeling power of OODBs. Illustra provides built-in extensibility of data types, so that we can easily define new abstract data types, such as ordered lists, as required by SGML modeling. Another advantage of Illustra is that it provides support for textual search functions through its "Text Datablade" package, supporting manipulation, storage and retrieval of free form, unstructured text of arbitrary length.

In addition, Illustra meets our project needs by supporting a mechanism for constraint specification and management. This is important since, for example, if we were to use this document database within an SGML editor or composer application, it would be necessary to have a mean to ensure that any object instance modified would still translate into a valid SGML document. Without these constraints being handled within the database, it would be necessary to translate an object back into a SGML document and run a parser on it to determine whether or not it is still valid.

DMS System Overview. As depicted in Figure 1, the DTD-mapper module of our DMS system translates an input DTD into our internal schema representation. Our schema-merger module merges the

newly generated classes with the integrated document schema. Given an SGML document instance, the loader module determines a match between the particular document and the integrated DTD schema. Based on this match, the document will be decomposed and loaded into the DMS repository as a complex document object. A query module is available to process extended SQL queries, while a graphical interface for displaying query results (in particular, structured document fragments) is under development.

3 The Constraint-Based Object Model

As the foundation of our system, we are assuming a general object model as adopted by most other OODBs [5, 6]. In order to support modeling of all SGML constructs, we extend the model with constraints over the attribute set. As we shall describe, the constraints are used to model properties of a subset of the attributes of a class and they must hold for every instance of the class.

Let N be the set of all classes, L the set of attribute names, and O the set of object instances. We represent a schema S as a tuple (C, E, G, T) where:

1. $C \subseteq N$ is a finite set of classes. For every class c in C, there is a collection of object instances in O that defines the content of the class c. The object instances are said to be members of (or instances of) their class c.

2. $E \subseteq C \times L \times C$ is a finite set of attributes. For an attribute $(p, a, q) \in E$, every instance of the class p has an attribute named a whose value is a member of the class q. The class q is called the domain of a for p. For all $e_i \in E$, such that $e_i = (p, a_i, q_i)$ for some class p, we call $\{a_i | (p, a_i, q_i) \in E , \forall a_i \in L, \forall q_i \in C\}$ the attribute set defined for p.

3. $G \subseteq C \times C$ is the specialization relation on classes. For $(p, q) \in G$, we say that p is a specialization of q (or q is the generalization of p) which indicates that every instance of p is also an instance of q. G is a reflexive, transitive and antisymmetric relation on C.

4. $T \subseteq (E \times E \times B) \cup (E \times U)$ is the set of binary and unary constraints over the attribute set E , where $B = \{b : E \times E \rightarrow \{True, False\}\}$ is a set of binary constraint functions and $U = \{u : E \rightarrow \{True, False\}\}$ is a set of unary constraint functions. T is defined as a finite set: $T = \{((p, a, q), (p, b, t), f) \mid f \in B , (p, a, q) \in E , (p, b, t) \in E$ and $f((p, a, q), (p, b, t)) = True\} \bigcup \{((p, a, q), g) \mid g \in U , (p, a, q) \in E$ and $g((p, a, q)) = True\}$.

   The set of constraints T is defined by enumerating the constraint tuples (argument(s), constraint function) corresponding to the $True$ values of the constraint functions. For the purpose of this paper, we use only binary and unary constraints over the attribute set of the same class. The data model could easily be extended with a more general set of constraints. However, we found this restricted set of constraint types sufficiently powerful to handle all SGML semantics.

An example of an unary function is the non-null constraint denoted by:

$NotNull : E \rightarrow \{True, False\}$.

This constraint is defined, for example, as follows for the class $Person$:

$NotNull((Person,\ SSN,\ Integer)) = True$

with the semantics that this is true if and only if any instance of the class $Person$ has an attribute SSN not null, with the later a member of the class $Integer$. This constraint assures that every person instance does have a legal ssn value.

An example of a binary function is the exclusive-or constraint denoted by:

$ExclusiveOr : E \times E \rightarrow \{True, False\}$.

This constraint is defined, for example, as follows for the class $Student$:

$ExclusiveOr((Student,\ VisaType,\ String),\ (Student,\ Citizen/PermRes,\ \ String)) = True$

with the semantics that every instance of the $Student$ class must have either an attribute value for its $VisaType$ or for its $Citizen/PermRes$ status, but cannot have values for both.

In addition, we establish that a schema $S = (C\ ,\ E\ ,\ G\ ,\ T\ )$ must satisfy the following properties in order to be considered well-defined:

1. If $(p, a, q_1) \in E$ and $(p, a, q_2) \in E$ then $\exists q \in C$ such that $(q, q_1) \in G$, $(q, q_2) \in G$ and $(p, a, q) \in E$.

   This property says that for every attribute a of a class p, there exists a *least* one class q that is subclass of all domains of a defined for a in p. Since all members of a class must meet *all* constraints defined on the class, this assures that there is ultimately one unique most restricted domain for each attribute of a class.

2. If $(p, q) \in G$ and $(q, a, r) \in E$ then $(p, a, r) \in E$.

   This property, often called the full-inheritance invariant, states that any attribute of q is inherited by all subclasses of q.

3. If $(p, a, q) \in E$ and $(q, r) \in G$ then $(p, a, r) \in E$.

   This property states that the attributes are preserved by the specialization relation over C.

4. (a) If $(p, q) \in G$ and $((q, a, r), g) \in T$ then for all $r' \in C$ for which $(p, a, r') \in E$ we have that $((p, a, r'), g) \in T$.

   (b) If $(p, q) \in G$ and $((q, a, r_1), (q, b, r_2), f) \in T$ then for all $r'_1, r'_2 \in C$ for which $(p, a, r'_1) \in E$, $(p, b, r'_2) \in E$ we have that $((p, a, r'_1), (p, b, r'_2), f) \in T$.

   The last property enforces that the constraints are also preserved by the specialization relation by being inherited from a generalization class to all its specialization subclasses. In particular, 4.a states the constraint preservation principle for unary constraint functions, and 4.b states it for binary constraint functions, respectively.

4 Mapping SGML Model Groups into the Constraint Based Model

In this section we define a general strategy for mapping SGML basic model groups to a constraint-based object schema based on the model defined above (Section 3). The proposed strategy is general in that it could easily be realized on any OODB system that supports the elementary OO modeling constructs, such as objects, classes, attributes, and generalization relationships. In addition, we assume that the target OODB system supports constraints. If this is not provided by the OODB system, then the constraints defined in the mapping process would have to be realized using method encoding.

The Example Document Type Definition (DTD). We will use the Document Type Definition (DTD) for documents of type *article* given in Figure 2 as source for the examples throughout the remainder of this section. This article DTD example was first defined in [7], and represents a realistic though simplified version of the Elsevier Science Article DTD documented by Elsevier Science Pub. [13]. In order to validate our approach, we also describe in this section how our DMS system realizes the resulting constraint-based object schema for the article document type using the Illustra DBMS engine [5].

```
 1. <!DOCTYPE article [
 2. <!ELEMENT article   -- (title, (author+), affil, abstract,
                                        (section+), (bib&ack))>
 3. <!ELEMENT title     -O (#PCDATA)>
 4. <!ELEMENT author    -O (#PCDATA)>
 5. <!ELEMENT affil     -O (#PCDATA)>
 6. <!ELEMENT abstract  -O (#PCDATA)>
 7. <!ELEMENT ack       -O (#PCDATA)>
 8. <!ELEMENT bib       -O (#PCDATA)>
 9. <!ELEMENT section   -O ((title, (body+))| (title, (body*),
                                                (subsectn+)))>
10. <!ELEMENT subsectn  -O (title, (body+))>
11. <!ELEMENT body      -O (figure|paragr)>
12. <!ELEMENT figure    -O (picture, (caption?))>
13. <!ELEMENT picture   -O (#PCDATA)>
14. <!ELEMENT caption   -O (#PCDATA)>
15. <!ELEMENT paragr    -O (#PCDATA)>
]
```

Figure 2: A DTD for a Document of Type Article.

The definition of the element **article** (line 2 in Figure 2) can be represented as a Directed Acyclic Graph (DAG) where the leaves are the elements of the SGML basic types (e.g., #PCDATA) and the internal nodes are SGML connectors (e.g., **"&"**, **","** and "|'), SGML indicators (e.g., **"?"**, **"*"** and **"+"**) and elements which are defined using other elements. Figure 3 depicts the DAG representation capturing the DTD from Figure 2.

General Mapping Strategy.

Step 1: Basic Types. In the general mapping, every SGML basic type (e.g., #PCDATA in Figure 2) is represented by a class of the appropriate type (e.g., **TEXT** or **DOC** in Illustra).

Step 2: Complex Types. Each basic SGML model group is represented by a class with its attributes corresponding to the elements of the model group and with the constraints imposed by the connector used in the model group. The name of the class is the name of the defined SGML element if the model group is the content of the element definition. For example, **(title, (body+))** is the model group defining the element **subsectn** in line 10 of the DTD in Figure 2 and it will be mapped into a new class with the name **SUBSECTN**. Note, for example, that in Figure 3, there exists a subtree representing this model group (T2) which is the only child of the node **subsectn**.

If the model group is used inside another model group then we will generate a unique name for the

[5] Recall that we have chosen Illustra since it supports both content-based queries as well as general constraints.

corresponding class. For example, the **(title, (body*), (subsectn+))** model group is used in the definition of the **section**. Thus a unique class name will be assigned to this model group so that it can be referred to in the **section** class. In this case, the subtree in Figure 3 corresponding to this model group (T1) is the child of a node labeled with the SGML connector "|".

Mapping Algorithm for Model Groups. Concrete mapping steps based on the SGML connectors and indicators are listed below:

Case 1: Optional Indicator. The optional indicator "**?**" indicates zero or one occurrence of an element. A model group **(a?)** will be mapped into an attribute **a** and no constraints: the attribute **a** can be null.

A definition of an element p in SGML:

```
<!ELEMENT p    -O (... (a?) ...)>
```

will be translated into a new class p with one of its attributes equal to $(p, a, q) \in E$. No unary constraints will be associated with the attribute a. q is the class corresponding to a's model group, with the later defined using this mapping algorithm.

Case 2: No Indicator. An element with no occurrence indicator must be constrained to be not null using the unary constraint function **NotNull** defined above. A definition of an element p in SGML:

```
<!ELEMENT p    -O (... a ...)>
```

will be translated into a new class p with one of its attributes equal to $(p, a, q) \in E$. We also impose the $NotNull$ unary constraint $((p, a, q), NotNull) \in T$. Again, q is the class corresponding to a's model group.

To illustrate the previous two mapping steps, we now give the translation of the **figure** definition from the DTD example (line 12 in Figure 2):

```
<!ELEMENT figure    -O (picture, (caption?))>.
```

In Illustra, a new complex class **FIGURE** and the corresponding table **FIGURE_table** are created as following:

```
create table FIGURE_table of new type FIGURE (
    picture DOC not null,
    caption DOC
);
```

Not-null is an unary constraint that is directly supported in Illustra. The constraints imposed by the sequence indicator "," for **(picture, (caption?))** model group, are defined below in Case 5.

Case 3: Closure-Occurrence Indicator. The asterisk sign indicator "*****" indicates zero or more occurrences of an element. The model group **(a*)** is translated into an attribute of ordered-list type with no unary constraint imposed: the list could be empty. The ordered-list type composed of elements of the same type a must be supported by the target OODB system.

A definition of an element p in SGML:

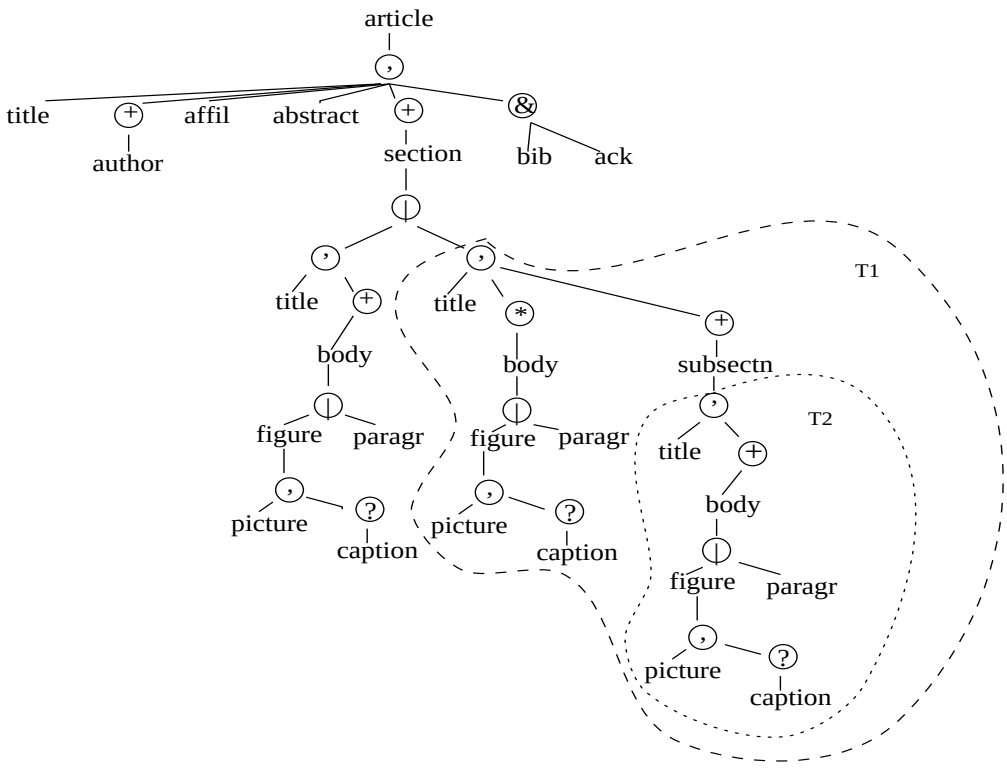

Figure 3: Directed Acyclic Graph Representation for the Element *article*.

```
<!ELEMENT p    -O (... (a*) ...)>
```

will be translated into a new class p with one of its attributes equal to $(p, a, q) \in E$. No unary constraints are specified for the corresponding attribute a. q is the ordered-list class with the element type corresponding to a's model group. Since Illustra does not support any built-in variable-length lists, we model the ordered list in Illustra by using a position field which gives the position of the elements in the list.

Case 4: Multiple-Occurrence Indicator. The plus sign indicator **"+"** indicates one or more occurrences of an element. The model group is translated like the **"*"** model group but with an unary $NotNull$ constraint imposed. A definition of an element p in SGML:

```
<!ELEMENT p    -O (... (a+) ...)>
```

will be translated into a new class p with one of its attributes equal to $(p, a, q) \in E$. In addition, we specify the unary constraint $((p, a, q), NotNull) \in T$. q is the ordered list class with the elements of type corresponding to a's model group.

In our example, the model group **(title, (body*), (subsectn+)))** (part of line 9 in Figure 2) is translated as following:

```
create table  A2_SECTION_table of new type A2_SECTION (
   title DOC not null,
   bodys BODY_list,
   subsectns SUBSECTN_list not null
);
```

with **BODY_list** and **SUBSECTN_list** defined separately as described above.

Case 5: Sequence Aggregation Connector. The SGML sequence (aggregation) connector "," imposes an order between elements connected by it. In the constraint-based object model, an aggregation model group $(a_1, a_2, \ldots, a_n)$ will be translated into a class t, with attributes $(t, a_1, q_1), (t, a_2, q_2), \ldots, (t, a_n, q_n) \in E$ where q_i is the class corresponding to the domain of the element a_i. If a_i is a SGML basic type (e.g., #PCDATA) then q_i is the class of the appropriate type (e.g., **DOC** in Illustra). If a_i is another model group, q_i is the class corresponding to a_i's model group.

In order to support the semantics of the aggregation model group, we need to introduce the binary constraint function "$<$" defined by:

$$<: E \times E \rightarrow \{True, False\}$$

where $<((p, a, q_1), (p, b, q_2)) = True$ if and only if the value of the attribute a is always before the value of the attribute b for every instance of class p.

In particular, for the class t defined above, we add the following constraints:

$$((t, a_i, q_i), (t, a_j, q_j), <) \in T$$

for all $1 \leq i < j \leq n$. Remember that T was defined as a set of *all* constraint tuples corresponding to the $True$ values of the constraint functions. We currently have no way to infer that if $(e_1, e_2, f) \in T$ and $(e_2, e_3, f) \in T$ then $(e_1, e_3, f) \in T$. Hence we must enumerate all the tuples for which f is $True$ even f is

a transitive function as in the case of "<" function. Once the model is extended with inferencing support, we can optimize the constraint representation by removing redundant constraint tuples whenever deducible.

In our example, the definition of the element **subsectn** (line 10 in Figure 2):

```
<!ELEMENT subsectn  -O (title, (body+))>
```

is translated to a new class **SUBSECTN** with two attributes **title** and **bodys** defined as following:

```
create table SUBSECTN_table of new type SUBSECTN (
   title DOC not null,
   bodys BODY_list not null
);
```

The constraints are supported by creating a rule which for example imposes the order of the attributes when the **select** is executed on **SUBSECTN_table**:

```
create rule select_SUBSECTN
on select to SUBSECTN_table
do instead
select title, bodys from
SUBSECTN_table ;
```

The constraints indicated by the aggregation connectors must be explicitly imposed for every operator that accesses a subset of the attributes of **SUBSECTN_table** in the case of the Illustra system. For example, we have defined a function **GetDocText** for every class in the schema which returns the SGML text of an article instance. The order in which the different pieces are composed into the structured document is imposed by the aggregation connectors used in defining sequence model groups and by the original order of the instances corresponding to "+", "*" and "&" model groups.

We do not introduce a rule for the insert operation, because we want to allow the user to dynamically insert or edit subcomponents of the document's subsections at their convenience. However, if a whole document instance is inserted at a time, we assume that the SGML instance loader (instance parser) will check the constraints imposed by the sequence model groups by utilizing the DTD as grammar.

Case 6: Alternative Aggregation Connector. The SGML alternative aggregation connector "&" imposes no order between elements connected by it. In the formal constraint-based object model, an alternative aggregation model group $(a_1 \& a_2 \& \ldots \& a_n)$ will be translated into a class t, with attributes (t, a_1, q_1), (t, a_2, q_2), $\ldots, (t, a_n, q_n) \in E$ and $n!$ subclasses $t_i \in C$, $(t_i, t) \in G$ for $1 \leq i \leq n!$[6]. A subclass t_i has no attributes and has the constraints corresponding to one of the permutations of the set $\{1, 2, \ldots, n\}$. More precisely, there exists a unique permutation $(i_1, i_2, \ldots, i_n)$ of the set $\{1, 2, \ldots, n\}$, so that for the subclass t_i, we have $((t_i, a_{i_j}, q_{i_j})(t_i, a_{i_k}, q_{i_k}), <) \in T$ for all $1 \leq j \leq k \leq n$. The q_i's are the classes corresponding to the model groups of the a_i's. While the class t may not have any direct instances, its instance set will be the union of the instance sets of its subclasses $t_1, t_2, \ldots, t_{n!}$.

In our example, the model group **(bib&ack)** (line 2 in Figure 2) will be translated as following:

[6] To optimize, we will only generate a subclass t_i if it is actually being used — i.e., if it contains at least one document instance.

```
create table BIB_ACK_table of new type BIB_ACK (
   bib DOC not null,
   ack DOC not null
);

create table BIB_ACK_12_table of new type BIB_ACK_12 (
) under BIB_ACK_table;

create rule select_BIB_ACK_12
on select to BIB_ACK_12_table
do instead
select bib, ack from
BIB_ACK_12_table ;


create table BIB_ACK_21_table of new type BIB_ACK_21 (
) under BIB_ACK_table;

create rule select_BIB_ACK_21
on select to BIB_ACK_21_table
do instead
select  ack, bib from
BIB_ACK_21_table ;
```

The table **BIB_ACK_table** has no direct elements of its own. This can be expressed in Illustra by creating a rule for preventing the direct insert operation:

```
create rule insert_BIB_ACK
on insert to BIB_ACK_table
do instead nothing;
```

It is important to note here that while the SGML alternative aggregation connector "**&**" does not impose any order between elements, our DMS system still keeps track of the original order in which document instance fragments were loaded into the DMS database to assure that the documents will be displayed in the same manner as before having been entered into DMS. If this requirement of preserving the initial (though, of course, optional) document fragment ordering can be relaxed, then the SGML alternative connector "**&**" would simply be mapped to the one class t without any constraints.

Like with the "**,**" connector, the constraints defined for "**&**" must be explicitly imposed to any operator where the order is important (for example, **select** operator). However, for the **&** connector, the membership of an element instance in one of the subclasses will implicitly define the constraints. For example, two instances of **BIB_ACK_table** are equal if and only if they are both instances of the same subclass (**BIB_ACK_12** or **BIB_ACK_21**) and the values of the attributes **ack** and **bib** are equal. This semantics could be used to define an equality operator for two document instances.

The obvious advantage of the above approach is that our solution preserves document input and output equivalence. Another advantage of this representation is that we can query attributes based on their position. The query *"Find all the articles where the acknowledgement precedes the bibliography"* will be translated into

a query which returns the articles for which the attribute **bib_ack** is an instance of the class **BIB_ACK_21** (the class with the constraint "**ack** is before **bib**").

Case 7: Choice Connector. The choice connector "|" provides an alternative for which element is to participate in the choice model group instance: $(a \mid b)$ indicates that either a or b is not null but not both. To map a choice model group $(a_1 \mid a_2 \mid \ldots \mid a_n)$ into our constraint-based model we create a new class $t \in C$ with no attributes and n subclasses $t_1, t_2, \ldots, t_n \in C$, $(t_i, t) \in G$ for all $1 \leq i \leq n$. A subclass t_i has only one attribute $(t_i, a_i, q_i) \in E$ and an unary constraint $((t_i, a_i, q_i), NotNull) \in T$ for all $1 \leq i \leq n$ where q_i is the class corresponding to a_i's model group.

Next, we explain how we would implement this model group in Illustra using the following example. For the **body** element definition (line 11 in Figure 2) we define:

```
create table BODY_table of new type BODY (
);

create table BODY_figure_table of new type BODY_figure (
   figure FIGURE not null
) under BODY_table;

create table BODY_paragr_table of new type BODY_paragr (
   paragr DOC not null
) under BODY_table;
```

The supertable **BODY_table** will have no direct instances of its own, but a query over it will return the rows of its two subtables, e.g., the elements of type **BODY**. This construction is one example that demonstrates how our approach allows us to easily specify queries ranging across diverse document types, without having to use special query constructs nor having to worry about which element types are contained in which particular DTD.

5 Schema Merging Using the Constraint Model

In the previous section, we have defined a general algorithm for mapping a DTD into a constraint-based schema. We now address the issue of how we can represent a collection of SGML documents $\{(DTD_i, file_i)\}_{1 \leq i \leq n}$ where every document has its own DTD. One solution, which appears to be commonly adopted by other researchers, is to map every DTD into a separate schema. This requires the underlying OODB system to support multiple classes with the same class name. Furthermore assuming, for example, that there are several DTDs defining the element **subsectn**, a query like *Q1: Find all the subsections of the documents containing the sentence "electrical engineering"* must be split into numerous subqueries over all the schemas that define the class **subsectn**.

An alternative solution, which we propose instead, is to have an integrated schema representing the collection of all DTDs. This process of DTD integration is similar to the problem of view or schema integration. Our strategy here is that the elements defined in different DTDs having the same name are mapped into the same class even when their definitions are slightly different[7].

[7] This is clearly a simplifying assumption that could be relaxed using several techniques typically applied for view schema integration. One, a table of synonyms could be utilized to identify additional classes that should be merged since they express the same semantics. Vice versa, human input could help to determine when same-named constructs represent distinct semantics and thus should not be merged

Let's take for example two DTDs: one with the subsection as defined in Figure 2:

```
S1: <!ELEMENT subsectn  -O (title, (body+))>
```

and the other one having the **subsectn** defined as:

```
S2: <!ELEMENT subsectn  -O (title, (body*), paragr)>.
```

Our goal is now to define a merged schema representing both DTDs. We thus must address how the two disparate **subsectn** definitions can be mapped to an integrated class definition meeting all constraints. Our solution to this problem is to introduce one general **SUBSECTN** class capturing the common elements and constraints of the two definitions. In our example, this **SUBSECTN** class would have the attributes **title** and **bodys** and the constraint "**title** is before **bodys**".

In addition, we need two subclasses of the class **SUBSECTN**, one **SUBSECTN_1** with no attributes but with the constraint *NotNull* for the attribute **bodys** and the second **SUBSECTN_2** with one attribute **paragr** and the constraints "**title** is before **paragr**" and "**bodys** is before **paragr**" corresponding to the definitions S1 and S2, respectively. The two subclasses represent the difference between the two definitions of the **SUBSECTN** concept in the two DTDs. The subsections of the SGML document instances following the first DTD will become instances of the **SUBSECTN_1** class and the subsections of the SGML document instances following the second DTD will be instances of the **SUBSECTN_2** class. But, of course, all of them will be instances of the class **SUBSECTN**. The query *Q1* from above will thus be able to directly access the extent (the set of the instances) of the integrated class **SUBSECTN**.

While above we described one example of our schema merging strategy, we now present the general algorithm for mapping a set of DTDs into the same schema. It consists of the following three steps:

1. Merge the set $\{DTD_i\}_{1 \leq i \leq n}$ into one DTD_0:
   (a) Every element E defined in some subset $\{DTD_{i_k}\}_{1 \leq i_k \leq n}$ of m DTDs with the definitions
   $\{$ <!ELEMENT E def_j $>$ $\}$ for $1 \leq j \leq m$

   is defined in DTD_0 as

   <!ELEMENT E $((def_1) \mid (def_2) \mid \ldots \mid (def_m))$ $>$.
   (b) Every element E which is defined only in one DTD has the same definition in DTD_0.
2. Map DTD_0 into the constraint-based object schema (as defined in Section 4).
3. Optimize the resulting schema.

Note that schema merging can be naturally achieved within the confines of our constraint model. As indicated above (step 3 of the merging algorithm), optimization of the generated schema is desirable. However, a detailed treatment of optimization is beyond the scope of this paper. Instead, we enumerate a few optimizations to give the reader a flavor of the type of optimizations that we are considering. One type of possible optimization includes reducing the number of defined classes. For example, for the nested model groups with occurrence indicators "*" and "+", we would not need to define a new class, rather we would only add a new attribute. The choice connector model group can be translated into a superclass with all the common attributes and constraints of the subclasses corresponding to the elements of the group. The above definition of the **SUBSECTN** class would be the result of such an optimization.

6 Related Work

There is a growing body of literature on proposals for developing structured document management systems by integrating database (DBMS) and information retrieval (IR) technologies [4, 8, 17, 15]. For example, Estrella [9] is one such system, an object-oriented extension of the relational DBMS Oracle extended with a predefined 'text' class. No structural document modeling using SGML is considered.

Another example is the work by Böhm, Neuhold and others [4, 3], who study the integration of the OODBMS VODAK with the IR system INQUIRY. While their system is also designed to handle SGML documents, they take the approach of defining special-purpose meta-classes for realizing specific SGML constructs. We, on the other hand, have proposed a constraint-based object model that provides a generic mechanism for document modeling. Most importantly, our model is designed to support the integration of multiple DTDs into one integrated system – sharing class definitions among different DTDs whenever possible – rather than maintaining a distinct schema for each separate DTD.

Blake et al. [2] propose a mixed system - combining a relational schema with SGML support - to query a collection of SGML files. The SGML files are kept in the file system, while a fixed set of attributes about the documents are stored in the database. In their system, each DTD is stored separately in a file as well, and is reprocessed each time a document is accessed and/or queried. Our DMS instead provides for full document management support by decomposing each SGML document into its individual document fragments as encoded by SGML and then managing the SGML document as a complex document object.

[7] proposed a mapping strategy from SGML into the O_2 database. This work requires an extension of the underlying OODB system, namely, the support of new kinds of meta-data types such as paths and union types. We take a different approach to the problem by utilizing a constraint-based model. [7] assumes the mapping of each DTD to a separate schema, while our work introduces the notion of schema integration for integrated document modeling.

[8] propose a system that couples the IR module INQUERY with the functional DBMS IRIS. Their DMBS schema is designed to model one particular document type only. The documents are stored highly redundantly, being stored both in the DBMS as well as in the IR system. The focus of their work is on object retrieval based on a probabilistic inference net model, while our paper address the DTD modeling and merging problem.

The constraint-based object model we present is an extension of the formal object model presented in [5] — namely, it is extended with general constraint specification. Similarly, the merging strategy we indicate in Section 5 is based on previous work on schema integration. Our main contribution here is again the focus on the integration of constraint-based schemata. As far as we know, the application of schema merging to complex document handling has not previously been studied.

7 Conclusions

It has been widely recognized that document management systems that effectively manage structured digital documents should integrate functionalities of both database and information retrieval technologies [4, 8, 17, 15, 9, 2]. Towards this end, we have developed a DMS system that can handle complex mixed queries incorporating both structural as well as content-based requests.

As a foundation of this system, we have introduced in this paper a formal constraint-based object model. Rather than requiring the extension of the underlying DBMS platform with special-purpose data types for

handling particular SGML-constructs, as proposed previously, we have developed a general algorithm for mapping any SGML construct into our formal constraint-based object model.

To validate our approach, we implemented a working prototype of the DMS system using our constraint-based approach. In particular, in this paper, we describe a detailed example of applying our proposed algorithm for mapping the article DTD to our formal constrained-based object model, and then to our implementation on Illustra. This experiment demonstrates the ease of realizing the proposed constraint-based object model using current OODB technology. Most importantly, we show that our model easily supports the merging of arbitrary DTDs into one integrated schema. To the best of our knowledge, this is the first work studying the application of schema integration techniques in the context of the document management problem. Our approach allows for the specification of queries across diverse document types, if desired, without requiring special syntactic query constructs or query processing techniques. Our approach towards structured document handling offers genericity and simplicity, enabling any existing OODB system to be quickly extended with structured document management support.

In the future, we plan to investigate strategies for optimizing integrated constraint-based schemata generated by our algorithm. We also plan to experiment with our prototype to evaluate its effectiveness in document retrieval. This includes the integration of the DMS system into UMDL – the digital library system being developed at the University of Michigan – as collection database to allow for experimentation with structured versus unstructured document search techniques over the same document sets. Finally, the issue of query optimization of mixed queries on structured documents remains a largely unsolved problem.

Acknowledgments. The authors wish to acknowledge Wu-Chang Feng, Priya Raman, and other students at the University of Michigan for their efforts in helping to implement the *document management system* prototype on Illustra. We also thank John Price-Wilken for introducing us to SGML, and all other members of the UMDL team for providing us with a supporting environment for digital library research.

References

[1] W.P. Birmingham, K.M. Drabenstott, C. O. Frost, A. J Warner, and K. Willis. "The University of Michigan Digital Library: This is not your father's library". *Proc. of Digital Libraries*, pages 53–60, 1994.

[2] G.E. Blake, M.P. Consens, P. Kilpelainen, P. Larsen, T. Snider, and F.W. Tompa. "Text/ relational database management systems: Harmonizing SQL and SGML". pages 267–279, March 1994.

[3] K. Böhm and K. Aberer. "Storing HyTime Documents in an Object-Oriented Database". *CIKM'94*, pages 26–33, 1994.

[4] K. Böhm, A. Mueller, and E. Neuhold. "Structured Document Handling - a Case for Integrating Databases and Information Retrieval". *CIKM'94*, pages 147–154, 1994.

[5] P. Buneman, S.Davidson, and A. Kosky. "Theoretical aspects of schema merging". *Advances in DB Technology- EDBT, 3rd International Conference on Extending DB Technology*, Proceedings:152–167, March 1992.

[6] R. G. G. Cattell and T. Atwood, editors. *"The Object Database Standard, ODMG-93"*. M. Kaufmann, 1993.

[7] V. Christophides, S.Abiteboul, S.Cluet, and M.Scoll. "From Structured Documents to Novel Query Facilities". *SIGMOD '94*, Proceedings:313–323, May 1994.

[8] W. B. Croft, A. Smith, and H. R. Turtle. "A loosely-coupled integration of a text retrieval system and an object-oriented database system". *15th. Ann. Int'l SIGIR Conference*, Proceedings:223–232, 1992.

[9] C. Damier and B. Defude. "The Document Management Component of a Multimedia Data Model". *11th Int. Conf. on Research and Development in IR*, pages 451–464, 1988.

[10] C.F. Goldfarb. *"The SGML Handbook"*. Clarendon Press, Oxford, 1991.

[11] ISO8879. "Information Processing-Text and Office Systems-Standard Generalized Markup Language (SGML)". 1986.

[12] A.J. Lee, E.A. Rundensteiner, and S.W. Thomas. "Active OODB System for Genome Map Assembly". *Information Systems, Special Issue on Scientific Databases, to appear Spring 1995.*

[13] M. Poppelier and H. Van Der Togt. "Documentation of the Elsevier Science Article DTD". April 1994.

[14] E.A. Rundensteiner, L. Bic, J. Gilbert, and M. Yin. "Set-Restricted Semantic Groupings". *IEEE Transaction on Data and Knowledge Engineering*, 6:193 – 204, April 1994.

[15] P. Schaeuble. "SPIDER: A multiuser information retrieval system for semi-structured and dynamic data". *ACM-SIGIR'93, Pittsburgh, PA*, June 1993.

[16] ICDE Workshop on Active DBMS Systems. "Research Issues in Active Databases". *ACM*, 1994.

[17] T.W. Yan and J. Annevelink. "Integrating a structured-text retrieval system with an OODB system". *Proceed. of 20th VLDB Conf., Santiago, Chile*, 1994.