オブジェクトリレーショナルモデルを用いたXML管理機構の実現

西岡秀一[†] 鬼塚真[†]

音楽配信や画像配信に代表されるようにインターネット上におけるコンテンツ流通は重要性を増しつつある．コンテンツ流通で利用されるデータは，コンテンツIDフォーラムが規定するDTDに妥当であるXMLになっていくものと考えられ，今後はこれらの大量のXMLを格納・検索する機構が必要になると考えられる．本稿では，コンテンツ流通を支援するXML管理機構を提案する．本機構の特徴は，DTDをオブジェクトリレーショナルモデルにマッピングしXMLを格納する機能と，SQLを拡張した問合せ文の検索結果をXML化する機能である．これらの機能をORDBMS上で実装し，OODBMSをベースとしたXML-DBMSと比較実験を行い，格納効率に関しては1.5〜1.8倍，検索性能に関しては1.3〜9.9倍，本提案のXML管理機構が優れることを確認した．

XML Data Management System Using Object Relational Model

Shuichi Nishioka[†] and Makoto Onizuka[†]

Content distribution, such as music or image delivery, is a major goal for the Internet. As a union of the distribution, the meta data for the content will be described in XML that conforms to the specific XML subset like content ID forum (cIDf) schema. Therefore, it is important to manage valid XML data and to offer good performance in terms of both storage size and query response time. This paper proposes an XML data management system for the content distribution system using object relational model. It has two important features: storing valid XML data by mapping DTD to object relational schema and converting a result of SQL query to an XML data. We implement these functions on our ORDBMS and compare it against XML-DBMS based OODBMS in terms of query performance and database size. The results prove that our DBMS is between 1.3 times and 9.9 times faster than the XML-DBMS and yields a smaller DB, between 1.5 times and 1.8 times.

1. はじめに

近年，インターネットの普及，デジタル技術の進展等の基盤技術の整備によって，ネットワーク経由でデジタル化されたコンテンツが流通しやすい条件が整いつつある．この環境を利用し，電子商取引・EDI（Electronic Data Interchange）等が実現されている．これらにおけるデータの流通単位はコンテンツ実体と当コンテンツに関する属性情報で構成され，属性情報はデータ構造を柔軟に記述できるXML（Extended Markup Language）[1]により，表現されつつある．

特にコンテンツ流通で利用されるXMLは，コンテンツ流通に関するフレームワークの提案団体であるDublin Core[2]やコンテンツIDフォーラム（cIDf）[3]の規定したDTD（Data Type Definition）に妥当なデータになっていくものと考えられる．これらの妥当なXMLが大量に流通すると，XMLを管理・操作することが必要である．このため，XMLを効率良く格納し，格納されたXMLに対し高速に検索を行うXML管理機構の実現が待たれている．

本稿では，これらの要求を実現するXML管理機構を提案する．本機構の特徴は，オブジェクトリレーショナルモデルを用いたマッピング手法により，妥当なXMLを格納し，SQLを拡張した問合せ文を用いて検索を行い，結果をXML化することである．

2章にて，妥当なXMLを格納するためのマッピング手法について利点・欠点を述べる．これらを基に，コンテンツ流通を支援するXML管理機構における課題を抽出し，必要機能についての検討を3章で行う．4章にて，これらの機能を実現したプロトタイプについて述べる．このプロトタイプの性能評価を5章で行い，本機構およびプロトタイプの有効性を確認する☆．

† NTTサイバースペース研究所
NTT Cyber Space Laboratories

☆ 以後，データを表現するXMLを「XMLデータ」，XMLにおける属性を「属性」，データベースにおける属性を「データベース属性」として記述する．

2. 関 連 研 究

本章では，妥当な XML データをデータベースに格納する手法について概説し，各々の利点・欠点について述べる．提案手法に関する詳細は，4 章で述べる．

妥当な XML データを格納する場合，3 つのマッピング手法[4]がある．

DTD 依存型 DTD のタグを，データベース属性やクラスに対応させてデータベーススキーマを決定し，そのスキーマに従って，XML データを分解して格納する手法

DTD 非依存型 DTD に依存せず DOM のクラス構成に対応させ，汎用のデータベーススキーマを決定し，それに従って，XML データを分解して格納する手法

全体型 clob/blob 型等を用いて，XML データをそのまま 1 つのカラム内に格納する手法

以下に，各手法に関する特徴を示す．

2.1 DTD 依存型

本手法における利点は，データベースが DTD に対応したスキーマ構成になるため，タグ情報がデータベーススキーマとして管理されることから，タグ情報は容量をほとんど必要としないことと，タグを意識した検索やタグ単位での更新に適していること等がある．一方，欠点は DTD の変更に対してスキーマを変更しなければならないことである．具体的な製品としては，インフォテリアの iConnector のようにサードベンダから提供されているものや，Oracle XDK のように DBMS ベンダが提供するものがある．これらの製品では DTD から，自動的にデータベーススキーマを生成することはできない．また，自動的にデータベーススキーマを生成する手法が提案されている[5),6)]が，XML とデータベースのデータモデル（リレーショナルモデル，オブジェクト指向モデル，オブジェクトリレーショナルモデル等）に相違があるため，いったん DTD を簡単化することにより，一部の情報が欠損する．本稿で提案する手法は，オブジェクトリレーショナルモデルを用いているため，データベース属性の順序がなく，同一属性名を複数格納できない．このため，DTD を簡略化しているが，コンテンツ流通においてはこの制約は問題にならない．

DTD をリレーショナルモデルへマッピングする手法は，文献 5) で提案されている．この手法（Shared Inlinig 法と Hybrid Inlining 法）は，DTD の簡単化を行い，リレーショナルスキーマを定義する．簡単化の基本操作は，要素型定義の内容モデルを展開する．具体的には，内容モデルの入れ子括弧を展開し，複数回出現する同一の要素を 1 つにまとめ，要素の出現順序情報を除く．この簡単化した DTD から JOIN を抑止するため，なるべく少ないリレーションを定義するが，いずれの手法を用いたとしても，簡易な検索を実行する場合に，JOIN が発生する可能性がある．提案手法では，JOIN をまったく行わず，トラバース処理を行うため，検索性能的に優位である．

XML の前身である SGML（Standard Generalized Markup Language）の DTD をオブジェクト指向モデルへ変換する手法は，文献 6) で提案されている．DTD の各要素を tuple，list，union の 3 タイプを基としたクラスへ定義している．提案手法により定義されるクラス数は，文献 6) の手法により定義されるクラス数に比べ，木構造モデルで表現される DTD における最下段の要素数だけ少ない．このことにより，トラバース回数を抑制することが可能となり，検索性能的に優位である．

2.2 DTD 非依存型

本手法における利点は，DTD を利用せずにデータベースに XML データを格納することから，XML データの構造が変更になっても開発者には負担がない．また，各要素（タグ，属性等）に従いデータが分解されて DBMS に格納されているため，前節の手法と同様，タグを意識した検索やタグ単位での更新に適している．しかし，前節の手法と比較した場合，DTD の構成を意識しない汎用な格納手法であることから，タグ情報も通常のデータと同様に格納することになるため，データベース空間をより多く必要とする傾向がある．具体的な製品としては，eXcelon や Tamino 等がある．5 章において提案手法と，この手法により XML を格納する XML DBMS とを比較する．

2.3 全 体 型

本手法における利点は，XML を分解せずそのまま格納することから，先の 2 手法と比較してデータの登録と検索結果の取り出しが高速に行える．また，XML データの構造が変更になっても特に開発者には負担がなく，通常は全文検索機能が提供され，製品によってはタグを意識した検索も可能である．しかし，XML データ全体が 1 つのカラムに格納されるため，タグ単位での更新のように XML データの一部を更新する操作は適さないという欠点がある．具体的な製品としては，Oracle interMedia や DB2 XML Extender，DB2 Text Extender 等がある．

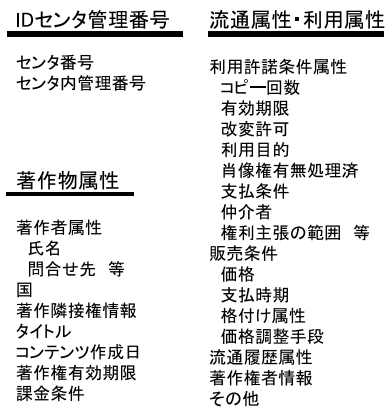

図 1 コンテンツ流通における属性例
Fig. 1 Sample tags for content distribution.

3. XML管理機構

本章では，コンテンツ流通を支援するXML管理機構の必要機能について述べ，それらの検討を行う．

3.1 必要機能

コンテンツ流通のフレームワークを提案しているcIDfは，コンテンツに関する属性情報を規定している．それらの属性情報は，**図1**に示す項目から成る（文献7)より抜粋）．図中の属性項目に関する変更はほとんどないため，DTDを用いて構造を規定することが可能である．このDTDに妥当なXMLを管理する場合，以下の機能が必要である．

機能1 XMLのタグごとに更新を行う（ノードの追加も含む）．コンテンツ流通物である原コンテンツに関して，契約や販売が行われる場合，図1における「流通属性」や「販売属性」等を更新する．

機能2 中間ノード等で複数回繰り返される構造を有するXMLを効率的に格納する．あるコンテンツに関して，多数の契約や販売が行われるため，「流通属性」や「販売属性」等が1つのコンテンツにつき複数付与される．

機能3 XMLに対する汎用で高速な検索を行う．属性情報に対して，複数の利用者から検索の要求が行われるため，これらを短時間で処理しなければならない．

3.2 機能検討

前節で述べた必要機能を基に，前章の3つのマッピング手法について検討を行う．

機能1を実現する場合，前章で述べたマッピング手法のうち，「全体型」はタグごとの更新ができないことから，採用困難である．よって，「DTD依存型」と「DTD非依存型」の手法について，以後検討を進める．

機能2を実現する場合，格納対象であるXMLの構造に対する保存能力が重要である．「DTD依存型」は，DTDで再帰的な構造として定義される要素（再帰的タグ）が存在する場合や，同名タグの出現順序に意味がある場合等は，XMLデータの格納時に，情報欠損が発生する．一方，「DTD非依存型」は，DTDによらず格納しているため，保存能力に長けている．図1をDTD表現する場合，再帰的タグは存在せず，繰返し構造の順序に意味がないため，いずれの格納手法を用いても，コンテンツ流通で用いるXMLデータの格納が可能である．

機能3を実現する場合，XMLに対する検索文の記述能力が重要である．「DTD依存型」は，問合せ言語としてSQL3を用いている．SQL3をベースにした問合せ言語は，スキーマを特定しないような検索（XQLで //，ancestor::，descendantや*を用いる場合）や要素間の順序（following-sibling，preceding-sibling）を一般的には表現が不可能である．しかしDTDが存在する場合は，DTDを用いることで要素間の順序がDTDにより決定され，かつスキーマを特定する検索に書きかえることが可能である☆．この結果，コンテンツ流通のようにDTDが存在する場合SQL3の拡張言語とXQLの表現能力は，両者とも同程度である．

機能2および機能3におけるXMLの格納効率とXML検索の高速性については，より有意なデータを取得するため，実用製品をベースとして比較を行う．具体的に，「DTD非依存型」については市販のOODBMSをベースとしたXML-DBMS，「DTD依存型」については筆者らが開発し，画像検索[9]や音楽検索[10]等に適用しているORDBMS LiteObject[11)~14)]を用いる．

4. XML対応ORDBMS

本章では，オブジェクトリレーショナルモデルを用いてXML対応機能を実現したORDBMS LiteObjectについて述べる．XML対応機能に関する概要を述べた後，DTDをオブジェクトリレーショナルスキーマへ変換する手法と，SQLによる問合せ結果をXML化する手法について詳細に述べる．

4.1 LiteObject XML対応機能の概要

LiteObjectのシステムアーキテクチャを，**図2**に示す．図中における *XML Object Relational mapping*モジュールの基本機能は，妥当なXMLデータの入力機能と，検索結果の出力機能である．

☆ 複数回繰り返されるタグについてはDBMSに格納する際にリスト型として管理することで要素間の順序を格納し，検索時に条件として指定することを可能としている．

入力機能は，自動的に DTD からオブジェクトリレーショナルスキーマを作成する．このスキーマを基に妥当な XML データからオブジェクトリレーショナルデータを作成し，導出したスキーマとデータを用いて，DB を構築する．オブジェクトリレーショナルスキーマの生成については，次節で述べる．

出力機能は，以下の特徴がある．

検索結果の XML 化 通常の DBMS の問合せ処理を拡張した構造保存機能[13)]を用いることにより，XML の検索において重要な入力 XML の一部の構造を再構成することなくそのまま出力 XML へ出力できる機構[15),16)]を実現している．

API データベースにアクセスするための拡張インタフェースと検索結果を DOM 木としてアクセスするための DOM API を有する．拡張インタフェースは，データベース接続管理機能，トランザクション管理機能，SQL の発行機能である．

以上の *XML Object Relational mapping*モジュールにおける入出力機能以外に，SQL による検索を高度化する機能がある．その機能を以下に示す．

全文検索 全文検索はユーザ定義関数により実現されている．これは，特に ANY（内容モデルが任意型である）要素に対する検索に有効である．

指定タグ配下抽出 XML データにおいて，指定したタグの部分配下における構造を検索結果として取得したい場合，対象となるタグをすべて列挙することはユーザにとって負担が大きいため，最上位タグを指定することでそのタグ配下の構造をすべて返却するユーザ定義関数を実現している．

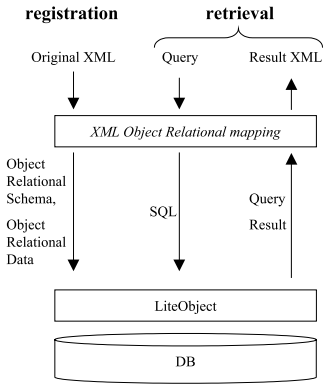

図 2 XML 対応 LiteObject アーキテクチャ
Fig. 2 LiteObject architecture for XML.

4.2 DTD のオブジェクトリレーショナルマッピング

本節では，DTD からオブジェクトリレーショナルスキーマを導出する方法について述べる．本手法は，簡単化した DTD より，要素の親子関係をリレーション間のキー・参照キーを用いて表現するのではなく，双方向関連を用いる．

図 3 に示す妥当な XML データを対象とした場合，リレーショナルモデルへのマッピング手法[5)]を参考に DTD の簡単化を行う．内容モデルの展開例を**図 4** に示す．

DTD を簡単化した後，木構造グラフ（DTD グラフ）を作成する．各 ELEMENT に対し，内容モデルの各要素と属性を子ノードとして木構造化する．図 3 の DTD を例としたグラフを**図 5** に示す．

DTD グラフを基に，オブジェクトリレーショナルスキーマを作成する．基本方針は，リーフノードはデータベース属性に，それ以外のノードは，クラスに変換する．以下,「ノードに関する入次数」と「ノードに関する子ノードの有無」によるオブジェクトリレーショナルスキーマの定義手法について述べる．なお，入次

```
<!ELEMENT book (title, authors, abst?, etc)>
<!ATTLIST book URL NMTOKEN >
<!ELEMENT etc ANY>
<!ELEMENT title (#PCDATA)>
<!ELEMENT authors (author+)>
<!ELEMENT author (name, address)>
<!ATTLIST author ref_society IDREFS
          fax NMTOKEN #IMPLIED
          email NMTOKENS #IMPLIED >
<!ELEMENT name (firstname?, lastname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT paper (title, abst, authors, society)>
<!ATTLIST paper URL NMTOKEN >
<!ELEMENT abst (#PCDATA)>
<!ELEMENT society (#PCDATA)>
<!ATTLIST society id ID #REQUIRED >
```

```
<book>
  <title>XML information</title>
  <authors>
    <author email="nishi@ntt.co.jp " ref_society="1">
      <name>
        <firstname>shuichi</firstname>
        <lastname>nishioka</lastname>
      </name>
      <address>Yokosuka Japan</address>
    </author>
    <author email="oni@ntt.co.jp" ref_society="1 3">
      <name>
        <firstname>makoto</firstname>
        <lastname>onizuka</lastname>
      </name>
      <address>Yokosuka Japan</address>
    </author>
  </authors>
</book>
```

図 3 妥当な XML データ
Fig. 3 Valid XML data.

数とは，木構造において上位のノードから参照されている数のことを指す．図 5 の場合，paper や book ノードの入次数は 0，URL や society ノードの入次数は 1，title や abst ノードの入次数は 2 である．

1. 入次数が「0」の場合，子ノードの有無にかかわらず，該ノードをクラス定義する．図 5 の場合，paper と book ノードが該当する．
2. 入次数が「1 以上」の場合，該ノードに関して子ノードの有無を確認する．図 5 の場合，paper と book 以外の全ノードが該当する．

2.1. 子ノードが 1 以上存在する場合，該ノードをクラス定義する．図 5 の場合，中間ノード（society や author 等）が該当する．
次に，定義したクラスへ，入次数分の参照型データベース属性を定義する．参照する型は，親ノードに相当するクラス型である．データベース属性名には，接頭辞「__up__」を付与する．図 5 の場合，society クラスに paper クラスへの参照型データベース属性を名称「__up__paper」で定義する．最後に，親ノードに相当するクラスに参照型データベース属性を定義する．ただし，入次の関係が「*」か「+」の場合は，参照のリスト型データベース属性である．参照する型は，いずれも該ノードに相当するクラス型である．図 5 の場合，paper クラスに society クラスへの参照型データベース属性を名称「society」で，authors クラスに author クラスを参照するリスト型データベース属性を名称「author」で定義する．

2.2. 子ノードが 0 の場合，該ノードをデータベース属性として，親ノードに相当するクラスすべてに定義する．図 5 の場合，リーフノード（URL や address 等）が該当する．これらは文字・数字型属性であるが，入次の関係が「*」か「+」の場合は，文字・数字のリスト型データベース属性となる．図 5 の場合，author クラスにデータベース属性を名称「address」で定義する．
ただし，該ノードが XML における属性の場合，「@」の接頭辞を付与し，データベース属性名とする．図 5 の場合，paper クラスにデータベース属性を名称「@URL」で定義する．また，IDREF 型の属性の場合は，親ノードに相当するクラスへ，ID 型の属性が定義された要素に相当するクラスへの参照型データベース属性を定義する．IDREFS

```
before      after
(x, y)? -> (x?, y?)
(x, y)* -> (x*, y*)
(x | y) -> (x?, y?)

x**  -> x*
x??  -> x?
x*?  -> x*
x?*  -> x*

(x, .., x,..)   -> (x*, ..)
(x?, .., x?,..) -> (x*, ..)
(x?, .., x*,..) -> (x*, ..)
(x*, .., x?,..) -> (x*, ..)
(x*, .., x*,..) -> (x*, ..)

-> : transformation
```

図 4 DTD の簡単化
Fig. 4 Simplifying DTD.

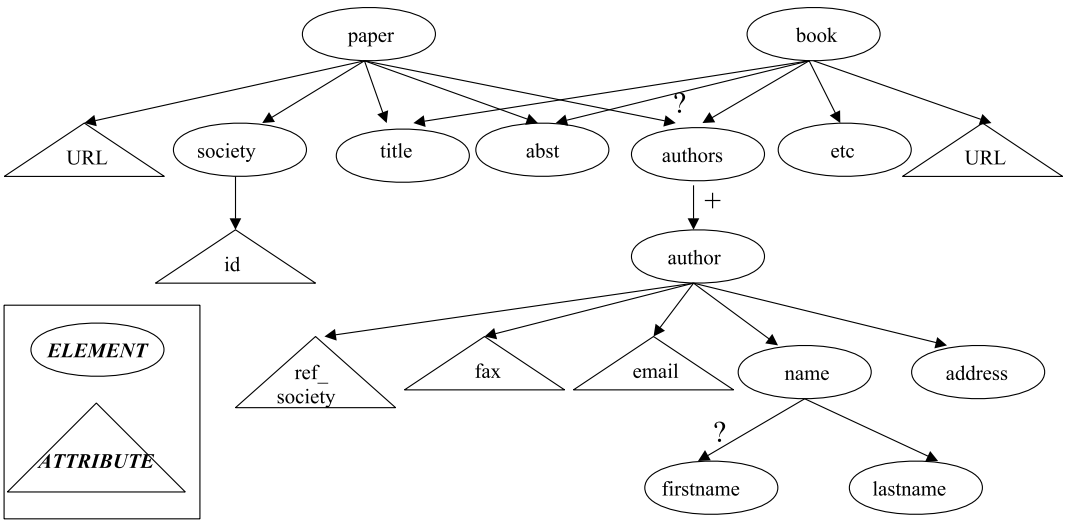

図 5 DTD グラフ
Fig. 5 DTD graph.

```
Class paper {
  string    @URL;
  society   society;
  string    title;
  string    abst;
  authors   authors;
}

Class book {
  string    title;
  string    abst;
  authors   authors;
  string    etc;
  string    @URL;
}

Class society {
  string    pcdata;
  string    @id;
  paper     __up__paper;
}

Class authors {
  List<author>   author;
  paper          __up__paper;
  book           __up__book;
}

Class author {
  List<society>   @ref_society;
  string          @fax;
  string          @email;
  name            name;
  string          address;
  authors         __up__authors;
}

Class name {
  string    firstname;
  string    lastname;
  author    __up__author;
}
```

図 6 オブジェクトリレーショナルスキーマ
Fig. 6 Object relational schema.

型の属性の場合は，参照のリスト型データベース属性である．図 5 の場合，author クラスに society クラスを参照するリスト型データベース属性を名称「@ref_society」で定義する．

上記手法により，DTD からオブジェクトリレーショナルスキーマが作成される．図 3 の DTD を例とした場合，**図 6** である．リーフノード以外でクラス定義を行っており，DTD グラフの木構造が保持されたクラス階層となっている．

4.3 検索結果の XML 化

本節では，図 3 に示す妥当な XML データを対象とした問合せ例を，SQL を用いて示す．また，問合せ結果を XML 化する手法についても述べる．

XML データに対して問合せを行う場合，SQL における検索対象，条件，返却したい構造等を以下の方法により表現する．

- 返却したいタグを，SELECT 句に列挙する．ただし，XML における属性を指定する場合は，XQL と同様に属性名に「@」を接頭辞として付与する．検索結果へ DTD の構造を保存させる場合は，列挙したタグを STRUCT ( ) で括る（SQL #1 参照）．また，STRUCT がない場合は，検索結果が平坦化される（SQL #2 参照）．
- 条件を指定するタグと具体的な条件を，WHERE 句に記述する．
- 返却したいタグと条件を指定するタグが参照可能である経路を検索対象として FROM 句に記述する．

上記の SQL により検索を行った結果を XML 化する場合，以下の項目を行う．

- ルートタグを result という名称のタグにする．
- SELECT 句で STRUCT が指定された場合，FROM 句で指定されたクラス群の関連階層に従い，タグの階層を構成にし，SELECT 句で指定されたタグを挿入する．本構成を検索結果の件数回繰り返し，検索結果の各値を SELECT 句で指定されたタグの値にする．
- SELECT 句に STRUCT が指定されていない場合，FROM 句で指定されたクラスを unnest 操作[11]し，平坦化したタグ構成を作成する．本構成を検索結果の件数回繰り返し，検索結果の各値を SELECT 句で指定されたタグの値にする．

```
<?xml version="1.0"?>
<result>
  <paper>
    <title>XML Object Relational Mapping</title>
    <authors>
      <author>
        <name>
          <lastname>nishioka</lastname>
        </name>
      </author>
      <author>
        <name>
          <lastname>onizuka</lastname>
        </name>
      </author>
    <authors>
  </paper>
  <paper>
    <title>survey about XML</title>
    <authors>
      <author>
        <name>
          <lastname>tanaka</lastname>
        </name>
      </author>
      <author>
        <name>
          <lastname>suzuki</lastname>
        </name>
      </author>
    <authors>
  </paper>
</result>
```

図 7 SQL #1 の検索結果（XML）
Fig. 7 XML output in response to SQL #1.

以下，問合せ例と検索結果の XML 例を示す．下記に示す SQL #1 は，

```
SQL #1:
SELECT STRUCT(a.title, d.lastname)
FROM paper a, a.authors b, b.author c, c.name d
WHERE a.title LIKE '%XML%'
AND c.address LIKE '%JAPAN%' ;
```

「論文のタイトルに「XML」という文字列を含み，著者が日本に住んでいるタイトルと著者の名字」という問合せである．検索結果は，**図 7** となり，簡単化した DTD の構造つまりオブジェクトリレーショナルス

```
<?xml version="1.0"?>
<result>
  <paper>
    <society>IPSJ</society>
    <firstname>shuichi</firstname>
    <lastname>nishioka</lastname>
  </paper>
  <paper>
    <society>ACM</society>
    <firstname>makoto</firstname>
    <lastname>onizuka</lastname>
  </paper>
</result>
```

図 8 SQL #2 の検索結果（XML）
Fig. 8 XML output in response to SQL #2.

キーマが保存されている．

次に，下記に示す SQL #2 は，

```
SQL #2:
SELECT a.society, d.firstname, d.lastname
FROM paper a, a.authors b, b.author c, c.name d
WHERE a.title LIKE '%XML%'
AND c.address LIKE '%JAPAN%' ;
```

「論文のタイトルに「XML」という文字列を含み，著者が日本に住んでいる論文の学会と著者の名前」という問合せである．検索結果は，**図 8** である．この XML の構造は，簡単化した DTD とは異なり，返却したいタグが平坦化されている．

最後に，下記に示す SQL #3 は，

```
SQL #3:
SELECT STRUCT(a.title, d.lastname)
FROM paper a, a.authors b, b.author c, c.name d
WHERE a.title LIKE '%XML%'
AND c.address LIKE '%JAPAN%'
ORDER BY 2;
```

「論文のタイトルに「XML」という文字列を含み，著者が日本に住んでいるタイトルと著者の名字」という問合せで，名字でソートしている．検索結果は，**図 9** となり，簡単化した DTD の構造と同様になるが，格納した XML（図 7）と比較すると，lastname タグにおける値の出現順序が異なっている．

以上より，LiteObject で処理可能な SQL では，簡単化した DTD の構造を保存した検索と，DTD と異なる構造へ変換（平坦化等）する検索が可能である．

```
<?xml version="1.0"?>
<result>
  <paper>
    <title>XML Object Relational Mapping</title>
    <authors>
      <author>
        <name>
          <lastname>nishioka</lastname>
        </name>
      </author>
      <author>
        <name>
          <lastname>onizuka</lastname>
        </name>
      </author>
    <authors>
  </paper>
  <paper>
    <title>survey about XML</title>
    <authors>
      <author>
        <name>
          <lastname>suzuki</lastname>
        </name>
      </author>
      <author>
        <name>
          <lastname>tanaka</lastname>
        </name>
      </author>
    <authors>
  </paper>
</result>
```

図 9 SQL #3 の検索結果（XML）
Fig. 9 XML output in response to SQL #3.

5. 評　　価

コンテンツ流通を支援する XML 管理機構は，複数クライアントから実行される検索等の要求を短時間に処理する必要があるため，高速な処理性能が求められる．

本章では，コンテンツ流通を想定したモデルを用い，OODBMS をベースとした XML-DBMS と XML 対応 LiteObject において，容量および検索に要した時間を測定した後，考察を述べる．

5.1 モ デ ル

図 1 に示すデータ構造を例とした XML データを 1,000 件，4,000 件，8,000 件を検索対象とした．タグの構造は，ルートタグからの最大段数を 8，1 件あたりの XML データサイズを約 3 k バイト，平均ノード数を 90 とした．なお，XML-DBMS では複数の XML データを対象とした場合，最初に格納した文書のみを検索対象とするため，格納用 XML データに新規タグ（root）を最上位に追加することで対処した．

先述の XML データを格納した 3 種の DB に対する検索モデルを，コンテンツ流通において頻繁な利用が予想される「author（著者名）タグの値が XXXXX であるタグ」とした．この問合せ文（SQL と XQL）

表 1 データサイズの比較
Table 1 Comparison with size.

データ数（件）	1,000	4,000	8,000
XML ファイル容量（MB）	3.5	13.7	28.0
XML-DBMS の DB 容量（MB）	12.0	49.0	100.0
LiteObject の DB 容量（MB）	14.6	32.4	54.6

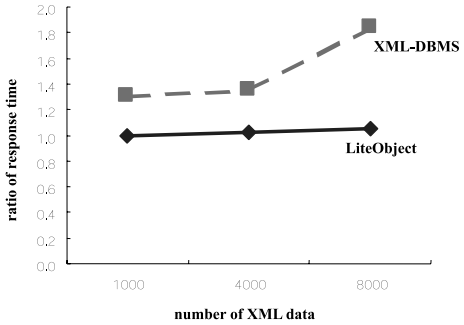

図 10 1 件検索におけるレスポンス時間比
Fig. 10 Ratio of response time relative to 1 match query.

を以下に示す.

SQL:
SELECT b.author
FROM ContentID a, a.contentInfo b
WHERE b.author = 'XXXXX';
XQL:
/root/ContentID/contentInfo/author[.='XXXXX']

5.2 測　　定

測定は，以下の環境で行った.

- PC（Pentium III 500MHz * 2）
- Windows NT 4.0 SP5
- Memory 1GB

XML データ 1,000 件，4,000 件，8,000 件に相当するファイル容量とデータベースに格納後のディスク容量を**表 1** に示す．各 DB 容量には，インデックス分は含まれていない.

表 1 における各 DB に対し，前節で示した問合せを行った．問合せの条件は，検索結果が格納件数の 1 件である場合，1%である場合，10%である場合の 3 通りとした．以後，これらの条件で行う検索を 1 件検索，1%検索，10%検索とする．測定した検索時間は，100 回実行した平均時間とした．**図 10**，**図 11**，**図 12** に LiteObject に格納した 1,000 件 DB に対して，検索に要した時間を 1 としたレスポンス時間比を示す．各 DB には，条件で指定したタグにインデックスが構築済みである.

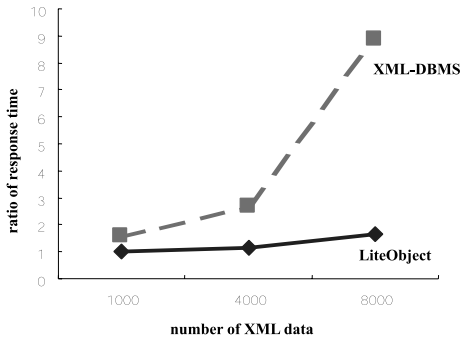

図 11 1%検索におけるレスポンス時間比
Fig. 11 Ratio of response time relative to 1% query.

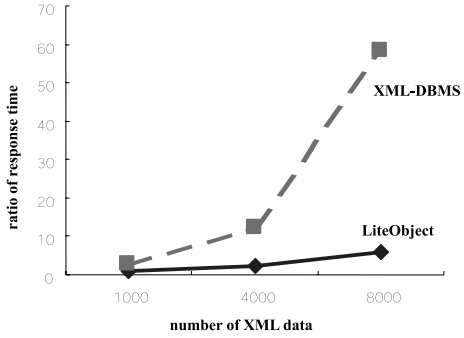

図 12 10%検索におけるレスポンス時間比
Fig. 12 Ratio of response time relative to 10% query.

5.3 考　　察

表 1 より，いずれの DBMS においても，データベースへ格納することで，ファイル管理の場合より，ディスク容量が増加する．XML-DBMS と LiteObject を比較すると，データ件数が 1,000 件の場合は XML-DBMS の容量が少なく，データ件数が 4,000 件および 8,000 件の場合は LiteObject が少ない．各 DBMS において，XML-DBMS は件数と比例して容量が増加するが，LiteObject は件数と比例せずに緩やかに増加する．このことにより，LiteObject では DB 作成時に XML-DBMS に比べるとディスク容量が若干要すると考えられる．また，XML-DBMS の容量に関する増加割合が LiteObject に比べて高い理由は，DTD に依存せず XML を格納するため，タグ情報に関して通常のデータと同様に格納することにより，データベース空間を要したと考えられる.

図 10，図 11，図 12 より，3 点の特性がある．1 点目は，3 種類の検索において，データ件数によらず，LiteObject の方が XML-DBMS より 1.3〜9.9 倍検索性能が良い．2 点目は，1 件検索における両者の性能差より，検索結果が多い場合の検索における性能差が拡大する．これらは，インデックスの性能，トラバー

スの回数によると考えられる．LiteObject では，値インデックスとして T 木を採用している．T 木は，文献 17) によると，B 木より高性能であると報告されている．トラバースの回数に関しては，クラスの定義数が異なることが要因として考えられる．XML-DBMS では DOM に近い形式でクラスを定義し XML を格納していると考えられる．このため，オブジェクトの粒度が LiteObject より細かくなることから，トラバース回数が増大したと考えられる．3 点目は，各検索において，いずれの DBMS も検索結果に該当する件数が増加すると，結果が少ない場合に比べ，性能に影響が出ている．これは，サーバ側で作成された返却結果に要するデータサイズが増加したため，通信量が多くなったことが要因として考えられる．

以上より，データベース容量・検索性能とも，オブジェクトリレーショナルモデルによるマッピング手法を実現した LiteObject の方が有効である．ただし，対象とした XML-DBMS へ大量データを格納するため，メモリ領域に関する設定の変更を行っている．この変更により，検索性能に影響が出ている可能性があると考えられる．また，XML-DBMS のベースとなっている OODBMS が用意しているインデックスの種類は，値インデックスのみであるが，パスインデックス等の導入により検索性能の差が縮小されると考えられる．

6. おわりに

本稿では，妥当な XML データをオブジェクトリレーショナルモデルへマッピングし DB に格納することと，格納したデータに対し構造保存検索等を用いて検索結果を XML 化することを特徴とする XML 管理機構を提案し，これらの機能を ORDBMS LiteObject を用いて実現した．また，この XML 管理機構について，OODBMS をベースとした XML-DBMS と比較を行い，格納効率および検索性能に関して優れていることを確認した．

今後は，本稿で実現した XML 管理機構における問合せ言語を XML 関連技術の言語に対応する．また，本管理機構を基に，コンテンツ流通の支援を行うシステムに関する研究開発を行う予定である．

参考文献

1) W3C: XML, http://www.w3c.org/xml/
2) Dublin Core Metadata Initiative, http://dublincore.org/
3) コンテンツ ID フォーラム：cIDf, http://www.cIDf.org/
4) 鬼塚 真：XML データベース開発者への 3 つの Q&A，*XML MAGAZINE*, Vol.10, No.8, pp.124–130 (2000).
5) Shanmugasundaram, J., Tufte, K., Zhang, C., He, G., DeWitt, D.J. and Naughton, J.F.: Relational Databases for Querying XML Documents: Limitations and Opportunities, *Proc. 25th International Conference on Very Large Data Bases* (*VLDB'99*), Sep. 7–10, Edinburgh, Scotland, UK, Atkinson, M.P., Orlowska, M.E., Valduriez, P., Zdonik, S.B. and Brodie, M.L.(Eds.), pp.302–314, Morgan Kaufmann (1999).
6) Christophides, V., Abiteboul, S., Cluet, S. and Scholl, M.: From Structured Documents to Novel Query Facilities, *Proc. 1994 ACM SIGMOD International Conference on Management of Data*, May 24–27, Minneapolis, Minnesota, Snodgrass, R.T. and Winslett, M.(Eds.), pp.313–324, ACM Press (1994).
7) 岸上順一，阪本秀樹：コンテンツ流通のビジネス動向，情報処理学会マルチメディア通信と分散処理 95-7, pp.37–42 (1999).
8) W3C: XQL, http://www.w3.org/TR/xql
9) 串間和彦，赤間浩樹，紺谷精一，山室雅司：色や形状等の表層的特徴量にもとづく画像内容検索技術，情報処理学会論文誌：データベース，Vol.40, No.SIG3 (TOD1), pp.171–184 (1999).
10) Kosugi, N., Nishihara, Y., Sakata, T., Yamamuro, M. and Kushima, K.: A Practical Query-By-Humming System for a Large Music Database, *Proc. 8th ACM International Conference on Multimedia*, Oct. 30–Nov. 4, Marina del Rey, CA, pp.333–342, ACM Press (2000).
11) Onizuka, M. and Okada, S.: Query Model for Structured Objects, *Proc. IFIP Int. Conf. on Database Semantics*, short paper, Jan., Rotorua, New Zealand, pp.32–45 (1999).
12) 岡田 敏，鬼塚 真，小西史和，谷口展郎，梅田昌義，小林伸幸，山室雅司：高速 ORDBMS LiteObject の設計と実装，電子情報通信学会第 9 回データ工学ワークショップ (1998).
13) 岡田 敏，鬼塚 真：画像検索アプリケーションにおける問い合わせ機構，情報処理学会研究報告 98-DBS-116-4, pp.25–32 (1998).
14) 西岡秀一，鬼塚 真，黒岩淳一，芳西 崇：ORDBMS（LiteObject）における複合メディア検索機構の実現，情報処理学会第 59 回全国大会 2Q-6 (1999).
15) W3C: XML Query Requirements, http://www.w3.org/TR/xmlquery-req
16) Levy, A.: More on Data Management for XML, http://www.cs.washington.edu/homes/alon/widom-response.html

17) Lehman, T.J. and Carey, M.J.: A Study of Index Structures for Main Memory Database Management Systems, *Proc. VLDB'86 Twelfth International Conference on Very Large Data Bases*, Aug. 25–28, Kyoto, Japan, Chu, W.W., Gardarin, G., Ohsuga, S. and Kambayashi, Y. (Eds.), pp.294–303, Morgan Kaufmann (1986).

(平成 12 年 12 月 22 日受付)
(平成 13 年 5 月 5 日採録)

（担当編集委員　中谷 多哉子）

西岡　秀一（正会員）

平成 7 年横浜国立大学工学部電子情報工学科卒業．同年日本電信電話株式会社入社．以来，オブジェクトリレーショナルデータベース管理システム，コンテンツ管理システムの研究開発に従事．

鬼塚　　真（正会員）

平成 3 年東京工業大学工学部情報工学科卒業．同年日本電信電話株式会社入社．以来，データベース設計，データ視覚化，オブジェクトリレーショナルデータベース管理システムの研究開発に従事．現在ワシントン州立大学客員研究員．ACM 会員．