データベースシステム　122－3
(2000. 7. 26)

XMLのオブジェクトリレーショナルマッピングに関する一手法

西岡 秀一, 鬼塚 真

NTT サイバースペース研究所
〒239-0847 神奈川県横須賀市光の丘 1-1
{nishioka, onizuka}@dq.isl.ntt.co.jp

あらまし

本稿では，情報交換の手段として注目されているXMLのうち，早期に流通すると予想される妥当なXML文書を対象とし，DTDを用いて，オブジェクトリレーショナルモデルへマッピングする手法を提案する．マッピングしたスキーマを基に，妥当なXML文書をオブジェクトリレーショナルデータへ変換し，ORDBMS (LiteObject) へ格納する．このDBに対し，SQLを用いてタグ構造を保存する検索と平坦化する検索を行い，結果をXML化して出力する手法も示す．また，提案する手法を用いて実現したORDBMSとOODBMSをベースとするXML-DBMSについて，データベース容量および検索性能について比較を行い，本手法により実現したDBMSが，格納効率が良く，性能が高速であることを確認した．

キーワード　XML，データベース，マッピング，ORDBMS

Mapping XML to Object Relational Model

Shuichi Nishioka, Makoto Onizuka

NTT Cyber Space Laboratories
1-1 Hikari-no-oka Yokosuka-Shi Kanagawa 239-0847 Japan
{nishioka, onizuka}@dq.isl.ntt.co.jp

Abstract

In this paper, we focus on valid XML documents as the best way of realizing information exchange. We propose an algorithm that map XML to the object relational model and that convert a result of SQL query to an XML document. There are two types of the result: structure preserving and flattened. We implemented our mapping algorithm on the ORDBMS LiteObject and compared with OODBMS based XML-DBMS in performance and database size. The result of this test proved the efficiency of LiteObject.

key words　XML, Database, Mapping, ORDBMS

1 はじめに

近年，ネットワーク環境が整備されたインターネット上で，デジタルコンテンツが流通し，電子商取引，EDI（Electronic Data Interchange）などが実現されている．このような流通形式として，柔軟なデータ表現能力を有する XML（eXtended Markup Language）[1] が注目されている．XML の出現により，コンテンツ ID フォーラム（cIDf）[2] のようなマルチメディアコンテンツの流通単位形式が表現可能となる．

XML で表現されたコンテンツが大量に流通すると，それらを格納・検索することが必要となり，大量の XML データを高速に操作可能な DBMS の実現が待たれる．

操作対象の XML には，整形式な文書（well-formed な文書）と妥当な文書（valid な文書）がある．整形式な文書はタグ構成を自由に定義できるが，妥当な文書は，DTD（Data Type Definition）によりタグの構成を定義している．

本稿では，XML のうち EDI などで早期に流通しやすいと考えられる妥当な XML 文書を対象としたオブジェクトリレーショナルモデルへのマッピング手法を提案する．また，マッピング後のデータを ORDBMS LiteObject[3] へ格納し，SQL にて検索を行った結果を XML 化する手法も示す．

次章にて，妥当な XML 文書のオブジェクトリレーショナルモデルへのマッピング手法と，結果を XML 化して出力する手法について述べる．次に，XML 文書を格納した LiteObject と，OODBMS をベースとした XML-DBMS について性能評価を行い，本手法の有効性を確認する．なお，XML 文書に対する問合せ言語として XQL[4], XML-QL[5] 等が提案されているが，今回は SQL による問合せを行った．問合せ言語は今後の課題となる．

1.1 関連研究

妥当な XML 文書を対象としたマッピング手法として，リレーショナルモデルへの手法 [6] やオブジェクト指向モデルへの手法 [7] が提案されている．また，整形式な XML 文書を対象とした DBMS の研究 [8, 9] も行われている．本稿では，オブジェクト指向ベースの DBMS との評価は行ったが，RDBMS との評価や異なるマッピング手法を用いた ORDBMS との評価は行っていない．これらは，今後の課題となる．

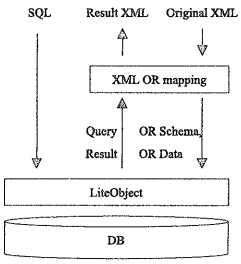

図 1: XML 対応 LiteObject アーキテクチャ

2 LiteObject の XML 対応機能

本章では，LiteObject の 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モデルへ変換する手法と，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[1].

2.1 XML 対応機能の概要

システムアーキテクチャを，図 1に示す．図 1における *XML OR 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．

入力機能は，DTD からオブジェクトリレーショナルスキーマを作成し，そ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する．変換後の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，後述する．

出力機能は，以下の特徴がある．

検索結果の XML 化 通常の DBMS の問合せ処理とは異なった構造保存機能 [10] を用いることにより，XML の検索において重要な入力 XML と出力 XML の構造が同様になる機能 [11, 12] を実現している．本機能により，検索にて指定したタグの構造化が可能となる．また，構造変換を行う検索も重要とされており，平坦化および

[1] 以後，DTD に妥当な XML の文書やデータを「XML データ」とし，「属性」は XML における属性，「データベース属性」はデータベースにおける属性として記述する．

```
<!ELEMENT paper (title, abst, authors, society)>
<!ATTLIST paper URL     NMTOKEN >
<!ELEMENT abst (#PCDATA)>
<!ELEMENT society (#PCDATA)>
<!ATTLIST society id ID #REQUIRED >
<!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)>
```

```
<?xml version="1.0"?>
<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 2: 妥当な XML データ

ソート機能で実現している．以上の機能については，後述する．

API DOM (Core) Level 1 仕様 [13] に定められているインターフェースの基本部分，および Document オブジェクト自体を取得するための LiteObject 独自のインターフェースから構成されている．独自インターフェースは，SQL を発行するための機能である．

以上の *XML OR mapping* モジュールにおける入出力機能以外に，SQL による検索結果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 DTD での任意要素である ANY により記述される自由な構造を有する XML の部分に対して，それらを一文字列として DB へ格納する．このことにより，ANY タグに対する検索は，全文検索ライブラリの利用により可能となる．

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のおける構造を検索結果として取得したい場合，対象となるタグを全て列挙することは困難であるため，一つのタグを指定することで，配下情報の構造を保持した検索が可能となるユーザーメソッドを実現している．

2.2 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．

```
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 3: DTD の簡単化

図 2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 [6] と同様に DTD の簡単化を行う．

簡単化の基本操作は，ELEMENT の内容モデルを展開し，グラフ化することである．内容モデルの展開では，以下の処理を行う．

- 入れ子括弧を展開する
- 同一参照要素をまとめる
- 要素の出現順序情報を除く

展開例を図 3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．図 2の DTD を例とした場合，図 4となる．

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

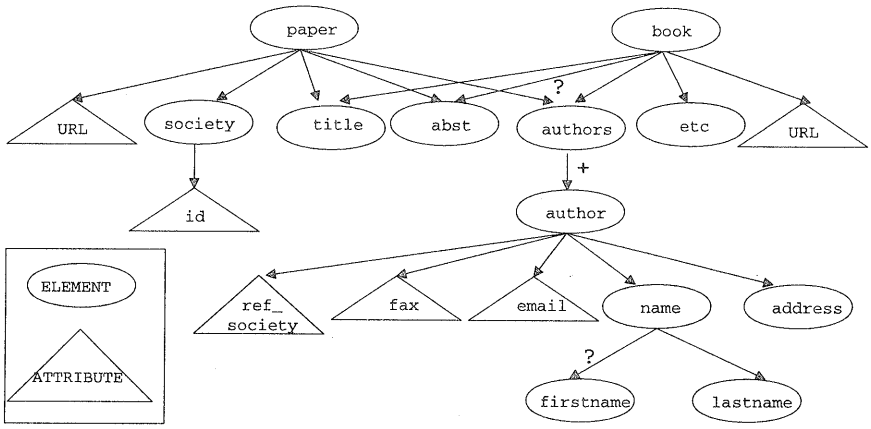

図 4: DTD グラフ

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．

1. 入次数が「0」の場合，子ノードの有無に関わらず，該ノードをクラス定義する．

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．

   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．定義したクラスには，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となる．また，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となる．参照する型は，いずれも該ノードに相当するクラス型となる．

   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全てに定義する．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．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 型の属性の場合は，参照のリスト型データベース属性となる．それ以外は文字，数字型属性となるが，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．

```
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     paper;
}

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

Class author {
 society   @ref_society;
 string    @fax;
 string    @email;
 name      name;
 string    address;
 authors   authors;
}

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

図 5: オブジェクトリレーショナルスキーマ

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 2の DTD を例とした場合，図 5となる．各ノード毎にクラス定義を行っているため，DTD グラフの木構造を保持したクラス階層となっている．

```
<?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 6: SQL #1 の検索結果 (XML)

2.3 検索結果の XML 化

本節では，図 2に示す DTD に妥当な 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，検索したい構造などを以下の項目により表現する．

- 取り出したいタグを，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 が無い場合は，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．
- 取り出したいタグと条件を指定するタグが参照可能となる経路を検索対象として FROM 句に記述する．

```
<?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 7: SQL #2 の検索結果 (XML)

上記の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をオブジェクトリレーショナルスキーマに従ったタグ構成にし，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 [14] 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．

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, d.name x
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 6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キーマが保存されている．

次に，下記に示す SQL #2 は，

```
<?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 8: SQL #3 の検索結果 (XML)

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，日本に住んでいる著者の所属学会と名前」という問合せである．検索結果は，図7となる．この XML の構造は，簡単化した DTD とは異なり，取り出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8となり，簡単化した DTD の構造と同様ではあるが，格納した XML(図 6) と比較すると，lastname タグの出現順序が異なっている．

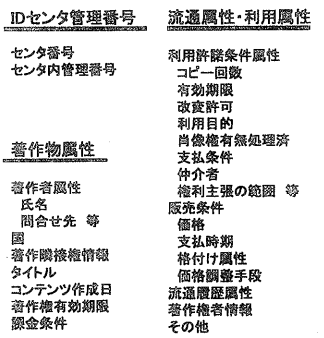

図 9: コンテンツ流通における属性例

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（平坦化など）する検索が可能となる．

3 評価

本章では，コンテンツ流通を想定したモデルを用い，市販 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．

3.1 モデル

図9に示すデータ構造（文献 [15] より抜粋）を例とした XML を 1000 件，4000 件，8000 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 3k バイト，平均ノード数を 90 とした．なお，LiteObject では複数の XML データを扱えるが，対象とした XML-DBMS では不可能であるため，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 3DB に対し，完全一致検索を 3 種類行った．各検索の一致率は，1 件（1 件検索），全件数の 1%（1%検索），全件数の 10%（10%検索）である．以下に，測定に用いた LiteObject SQL と XQL を

データ数 (件)	1000	4000	8000
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

表 1: データサイズの比較

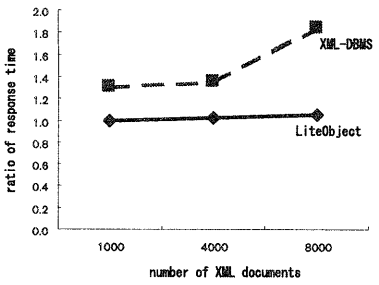

図 10: 1 件検索におけるレスポンス時間比

示す．いずれも「title タグの値が XXXXX であるタグ」を検索している．

LiteObject SQL:
SELECT b.title
FROM ContentID a, a.contentInfo b
WHERE b.title = 'XXXXX';

XQL:
/root/ContentID/contentInfo/title[.='XXXXX']

3.2 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 1000 件，4000 件，8000 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．

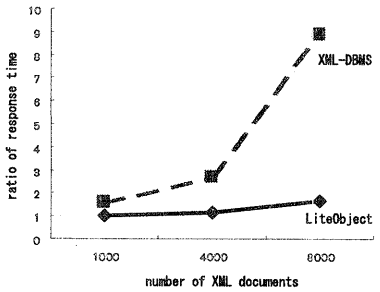

図 11: 1%検索におけるレスポンス時間比

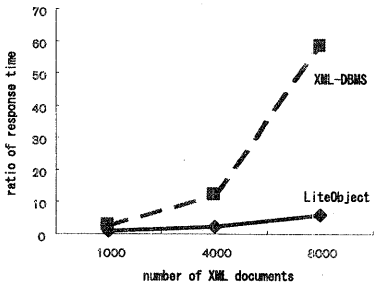

図 12: 10%検索におけるレスポンス時間比

表 1における各 DB に対し，前節で述べた 3 種類の検索を行った．LiteObject の 1000 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，図 10，図 11，図 12に示す．各 DB には，条件で指定したタグにインデックスが構築済である．

3.3 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，LiteObject が有効であると考えられる．

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 ～ 58 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．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，サーバー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，本稿にて提案した手法を実現した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．

4 おわりに

本稿では，妥当なXMLデータを対象とした場合のオブジェクトリレーショナルモデルへのマッピング手法を示した．同時に，オブジェクトリレーショナルモデルへ変換したデータに対して，構造保存検索が可能なORBMS LiteObjectを用い，検索結果をXML化する手法も示した．また，これらの手法を用いて，DBを構築し，データベース容量，検索性能において私達が提案した手法を実現したLiteObjectが有効であることを確認した．今後は，XMLに適した問合せ言語への対応を行う．

参考文献

[1] http://www.w3c.org/xml

[2] http://www.cIDf.org/

[3] “高速ORDBMS LiteObjectの設計と実装”, 岡田, 鬼塚, 小西, 谷口, 梅田, 小林, 山室, 信学会/第9回データ工学ワークショップ, 1998.

[4] http://www.w3.org/TR/xql

[5] http://www.w3.org/TR/xml-ql

[6] Jayavel Shanmugasundaram, H. Gang, Kristin Tufte, Chun Zhang, David J. DeWitt and Jeffrey F. Naughton, Relational Databases for Querying XML Documents: Limitation and Opportunities, In Malcolm P. Atkinson, Maria E. Orlowska, Patrick Valduriez, Stanley B. Zdonik and Michaeal L. Brodie, editors, VLDB '99, Proceedings of 25th International Conference on Very Large DataBases, September 7-10, 1999, Edinburgh, Scotland, UK, pp.302-314. Morgan Kaufmann, 1999.

[7] “From Structured Documents to Novel Query Facilities”, V. Christophides and S. Abiteboul and S. Cluet and M. Scholl, Proceedings of 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May, 1994.

[8] Daniela Florescu and Donald Kossmann, “Storing and Querying XML Data using an RDBMS”, IEEE Data Engineering Bulletin, Vol.22, No.3, pp.27-34, September 1999.

[9] Takeyuki Shimura, Masatoshi Yoshikawa and Shunsuke Uemura, “Storage and Retrieval of XML Documents using Object-Relational Databases”, In Proc. of the 10th International Conference on Database and Expert Systems Applications(DEXA'99), Vol.1677 of Lecture Notes in Computer Science, pp.206-217, SpringerVerlag, August-September 1999.

[10] “画像検索アプリケーションにおける問い合わせ機構”, 岡田, 鬼塚, 情処研報 98-DBS-116-4, pp.25-32, 1998.

[11] http://www.w3.org/TR/xmlquery-req

[12] http://www.cs.washington.edu/homes/alon/widom-response.html

[13] http://www.w3.org/TR/REC-DOM-Level-1/

[14] Makoto Onizuka and Satoshi Okada, “Query Model for Structured Objects”, short paper Proc. of IFIP Int. Conf. on Database Semantics, pp.32–45, January 1999.

[15] “コンテンツ流通のビジネス動向”, 岸上 順一, 阪本 秀樹, 情報処理学会 マルチメディア通信と分散処理, pp.37-42, 95-7, 1999.