博物館情報横断検索のための記述構造相互変換のプロトタイピングと評価

画像電子学会 VMA 研究会 博物館・美術館 DTD-SG
山田 篤 (京都高度技術研究所)　安達 文夫 (国立歴史民俗博物館)　海田 茂 (オープンアカウント)
今門 政記、河合 正樹 (インフォコム)　小町 祐史 (松下電送システム)

1. はじめに

画像電子学会 VMA 研究会「博物館・美術館文書の文書型定義 SG (DTD-SG)」では、博物館、美術館が公開する文書情報の横断検索について検討している。本研究グループが提唱する

I)　情報記述構造レベル
II)　情報記述内容レベル
III)　情報ナビゲーションレベル

からなる 3 層フレームワークについては[1][2]で報告した。

本稿ではこのフレームワークの実現に向けた考察の第一歩として、レベルⅠに関するプロトタイピングとその評価について述べる。我々の提唱するフレームワークでは各館の独自性、多様性を許容するため、様々なレベルでの相互変換を行う。このうち、レベルⅠでは文書記述の構造レベルでの相互変換を行う。

2. 記述構造の相互変換

2.1. 相互変換の基本的な考え方

記述構造の相互変換とは、たとえばデータベースのスキーマレベルで、A 館の x というスロットは B 館の y というスロットに相当するといった対応をとることによって、それぞれの館が独自に作成、管理している文書情報の横断検索を可能にするものである。

もしも、この記述構造が完全に標準化され、すべての館が統一された唯一の記述構造を採用するようになれば、このような相互変換は不要になるが、実際にはそのような標準化の実現は博物館・美術館の独自性・多様性という点からも難しいと考えている。一方で、記述対象の限定などによってそのような標準化の努力も必要と考えられる。

また仮に、一定の記述範囲での標準化が達成されたとしても、各館毎のインハウスのデータベースがなくなるとは考えにくい。その場合は、各館が独自に保有するデータベースの情報を標準化された構造にマッピングという形で変換することになる。

今、A 館と B 館の 2 館しかない状況で相互変換を考えると、それぞれの館から見て相手の館の情報が自館の記述構造に変換できればよい。すなわち、B 館の情報が A 館の記述構造に、逆に A 館の情報が B 館の記述構造に変換できれば相互に相手の館の情報を自館の情報とシームレスに横断検索できるようになる。これが相互変換の基本的な考え方である。

2.2. 横断検索のための構造変換ターゲット

このように2館しかない状況では変換は2とおりですむが、これがｎ館になると、最大で n(n-1)とおりの変換を要することは容易にわかる。さらに、横断検索ということを考えた場合は、ｎ館の情報が同じ構造で扱えたほうが望ましい。そこで、2館毎にそれぞれの変換を行うのではなく、横断検索用の共通の構造を中心としたスター型の構成をとることが考えられる。

博物館情報の横断検索に適した記述構造とはどのようなものかについては今後検討を要するが、本稿では試験的に Dublin Core Metadata Initiative (DCMI)によって策定されている Dublin Core Metadata Element Set V1.1 (DCMES)[3]を構造変換のターゲットとして設定した。DCMES で記述される項目を図 1 に示す。

1. title
2. creator
3. subject
4. description
5. publisher
6. contributor
7. date
8. type
9. format
10. identifier
11. source
12. language
13. relation
14. coverage
15. rights

図 1　Dublin Core Metadata Element Set

2.3. 横断検索対象データの作成

次にプロトタイピングの横断検索実験の対象となる文書データを作成した。このために、博物館・美術館 DTD-SG のメンバ6名がそれぞれ独立にウェブ上の仮想的な博物館を作成し、それぞれの館蔵品に対して独自にメタデータを付与した。これらのメタデータ作成時には基本的に DCMES は特に意識せず、それぞれの館蔵品やコレクションの性質や所蔵者の意図を反映したデータが付与されている。

3. プロトタイピング

3.1. 基本設計

記述構造としてDCMESを用いた複数博物館情報の横断検索の基本設計について述べる。

(1) DCMESデータの保管場所としてレポジトリを準備する。レポジトリは複数館で共通の登録場所としてもよいし、一つの館で一つのレポジトリを構成してもよい。レポジトリには各館独自のデータをDCMESに変換した結果を登録しておく。

(2) レポジトリとは別に検索サーバを設ける。検索サーバは各レポジトリの位置を把握している。ユーザからの検索条件が入力されると、検索サーバから各レポジトリに対して検索リクエストが投げられる。

(3)各レポジトリは検索サーバから送られた検索リクエストに対して検索結果を返す。

(4) 検索サーバは各レポジトリから返ってきた結果を表示する。

この構成を図2に示す。

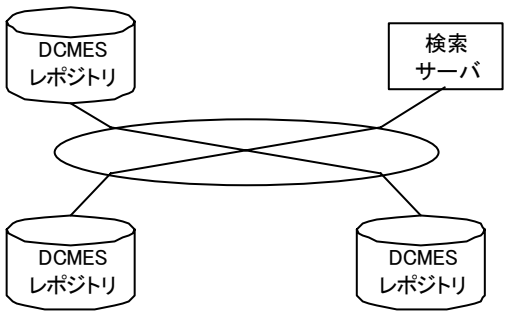

図 2 横断検索システムの基本構成

この場合、レポジトリにはDCMESに変換されたデータが格納されており、各館独自のデータ構造からDCMESへの構造変換はレポジトリへの格納前になされる。

3.2. DCMESへの変換

インハウスのデータ構造から DCMES のデータを作成する方法としては様々なものが考えられるが、ここでは構造変換という考え方にたち、あるデータ構造（ソース）から別のデータ構造（ターゲット）に変換するという一般的な仕組みを考え、ターゲットにたまたま DCMES を用いるという構成をとった。具体的には、XML ベースの記述方式を採用し、XSLT[4]を用いて構造変換を実現する。このとき、処理の流れは以下のようになる。

(1) 各館毎にインハウスの XML インスタンスを準備

(2) 各館毎にターゲットへの変換表を準備

(3) 変換表を用いてインハウスのXMLインスタンスをターゲットの XML インスタンスに変換

インハウスのXMLインスタンスは変換のソースとなるもので、各館独自のタグを用いて記述される。一方、ターゲットで用いるタグセットは共通のものとなる。DCMES の場合は、RDF/XML を用いた記述方法[5]が DCMI によって定められており、今回はそれを採用した。

ソースからターゲットへの変換時の対応関係は変換表として準備する。XSLT による変換時に読み込めるように、変換表も XML 形式で記述する。記述例を図 3 に示す。

```
<?xml version="1.0"?>
<correspondence>
 <tag>
  <to>dc:title</to>
  <org>title</org>
 </tag>
 <tag>
  <to>dc:creator</to>
  <org>creator</org>
  <org>made</org>
 </tag>
 <tag>
  <to>dc:subject</to>
 </tag>
.....
</correspondence>
```

図 3 変換表の記述例

変換表には、ターゲットの各タグに対しソースのどのタグが対応するかという情報を記述する。例の一つめの tag 要素は、ソースの title タグがターゲットの dc:target タグに対応することを示している。二つめの tag 要素はターゲットの dc:creator タグに対しソースでは creator と made の二つのタグが対応することを示す。三つめの tag 要素は、ターゲットの dc:subject に対して対応するソースのタグ(情報)が存在しないことを意味している。

インハウスのXMLインスタンスと変換表を読み込んでターゲットのDCMESへの変換を行うXSLT記述を図4に示す。このような単純な XSLT 記述で任意の変換が可能となる。この記述ではソースのインハウス XML インスタンスが 1 オブジェクトにつき、1 ファイルという構成を仮定しているが、複数オブジェクトが一つの XML ファイルに含まれている場合、XSLT プロセッサに固有の拡張機能（たとえば Xalan[6]の Redirect 機能）を利用すればターゲットを 1 オブジェクト毎の記述に分割することも可能である。変換結果の DCMES の例を図 5 に示す。

3.3. レポジトリの実現

このように作成したDCMES記述をレポジトリに格納する。レポジトリの実現方式としても様々な方法が考えられるが、検索サーバからの検索要求を処理する必要性からもデータベースシステムの利用が望ましい。先に DCMES を XML 形式で作成したが、DCMES の XML インスタンスは複雑な構

```
<?xml version="1.0"?>
<xsl:stylesheet
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
 xmlns:dc ="http://purl.org/dc/elements/1.1/"
 version="1.0"
 xml:lang="ja">
  <xsl:output method="xml" encoding="UTF-8"/>
  <xsl:variable name="trns_rule" select="document
(concat('cor_', /object_list/@owner, '.xml'))/
correspondence"/>
  <xsl:template match="/">
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match="object_list">
    <rdf:RDF xmlns:rdf="http://www.w3.
org/1999/02/22-rdf-syntax-ns#"
             xmlns:dc ="http://purl.org/dc/
elements/1.1/">
      <xsl:apply-templates/>
    </rdf:RDF>
  </xsl:template>
  <xsl:template match="object">
    <rdf:Description rdf:about="http://dublincore.
org/">
      <xsl:for-each select="child::*">
        <xsl:if test='string(.)'>
          <xsl:variable name="n" select="name()"/>
          <xsl:variable name="tag"
select="$trns_rule/tag[org/text() = $n]/to"/>
          <xsl:if test='$tag'>
            <xsl:element name="{$tag}">
              <xsl:value-of select="."/>
            </xsl:element>
          </xsl:if>
        </xsl:if>
      </xsl:for-each>
    </rdf:Description>
  </xsl:template>
</xsl:stylesheet>
```

図 4　DCMES への変換のための XSLT 記述

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#" xmlns:dc="http://purl.org/dc/elements/1.1/">
<rdf:Description rdf:about="http://dublincore.org/">
 <dc:identifier>01</dc:identifier>
 <dc:title>スヌーピー香立</dc:title>
 <dc:creator>不明</dc:creator>
 <dc:creator>中国</dc:creator>
 <dc:publisher>不明</dc:publisher>
 <dc:type>陶器</dc:type>
 <dc:date>2001.4.10</dc:date>
 <dc:rights>UFS</dc:rights>
 <dc:description>スヌーピー、ウッドストック</dc:description>
 <dc:description>香立</dc:description>
 <dc:description>ウッドストックの背中の穴にスティック状の香
をたてて使用</dc:description>
</rdf:Description>
</rdf:RDF>
```

図 5　DCMES への変換例

造をもたずフラットな形式になっているため、展開して RDB のテーブルに格納することも考えられる。しかしながら、将来的にこの記述がより複雑化したり、XML 形式のまま取り扱いたいといった要求が出てくることも考慮に入れて、今回は実験的に XML ネイティブなデータベースである Xindice[7]を用いた実装を行った。Xindice に対しては XML:DB API[8]を用いて接続する。検索には XPath[9]を、DB 更新には XUpdate[10]を用いる。

先に変換した DCMES の XML インスタンスを Xindice に登録してレポジトリは完成である。

3.4.　検索サーバの実現

検索サーバはユーザからの検索条件の入力を受け付け、レポジトリに対してリクエストを送り、レポジトリから返ってきた検索結果をユーザに提示するという一連の処理を実行するネットワーク上の仕組みであり、一般的にはウェブサーバ上の何らかのプログラムとして実現することになる。今回はラピッドプロトタイピングという要請から XML コマンド Xi[11]を用いて実現した。これに伴い、ウェブサーバとしては BayServer[12]を用いている。

検索方法としては DCMES のタグセットを用いた検索（図 6）と、データのどこかに検索語が入っていればヒットするフリーフォーマット検索（図 7）の 2 種類を準備した。検索結果の表示例を図 8 に示す。この検索結果からどの館のどの ID をもつ館蔵品かがわかるので、これらをキーにして、オリジナルの館のページにリンクすることもできる。

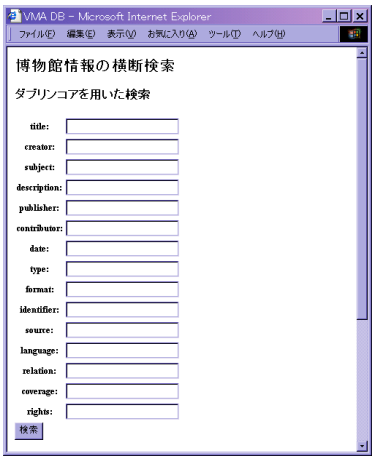

図 6　DCMES を用いた検索画面

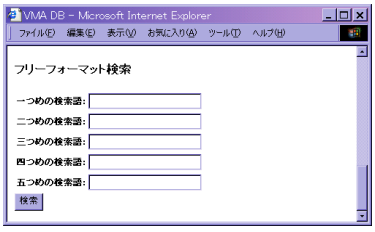

図 7　フリーフォーマット検索画面

4.　考察

4.1.　DCMES に関する考察

今回、相互変換の共通ターゲットとして DCMES を用いたが、はたしてこれが妥当なものであったかという問題がある。実際に各館独自のデータから DCMES への変換表を作成する際に、どのようなマッピングをすればよいのか戸惑った場合も少なくなかった。DCMES はやはり書誌情報の記述に最もフィットするように設計されていると

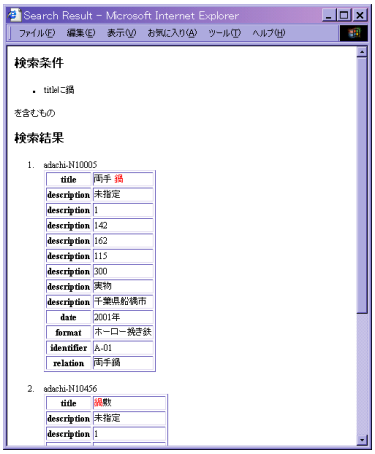

図 8　検索結果の表示例

いう感想も聞かれた。

また、ソースの複数の情報が DCMES の同一のエレメントにマッピングされるような場合もあり、それらは例えば Dublin Core の qualifier 等の仕組みを使わなければ区別できない。博物館情報の横断検索に適した変換ターゲットの設計は今後更なる検討を要する課題である。

4.2.　記述構造の変換に関する考察

何らかの共通ターゲットができたとしても、各館が独自に保持するデータから変換を行う際に、共通ターゲットの使い方に関して何らかのガイドラインが必要となることが考えられる。横断検索時に共通ターゲットの記述構造がうまく機能するためには、それらが統一的な用いられ方をしている必要があるだろう。このとき、共通ターゲットは DCMES のような完全なメタデータ形式でもよいが、ソースの情報は自由記述へのアノテーション形式のものの使用も考慮に入れるべきであると考える。その場合も変換という考え方は適用可能である。

また、今回のプロトタイピングではレベル I の情報記述構造レベルの相互変換のみを対象としたため、DCMES を用いたデータの構造化はなされているものの、そのコンテントについては文字列マッチングを行ったに過ぎず、その意味内容については一切扱っていない。これらはレベル II 以降に関する検討の過程で取り扱う予定である。

4.3.　実装方式に関する考察

今回の基本設計では記述構造変換後のデータをレポジトリに集める構成をとったが、この変換をいつ誰が行うかについては、いくつかの可能性がある。

一方にインハウスのデータベースがあり、他方に今回設計した検索サーバがあるような状況を考えてみる。要件としては検索サーバからは DCMES の構造に基づく検索要求が出され、DCMES の構造の検索結果を受け取るという条件があるだけである。よって、検索サーバとインハウスのデータベースの間に何らかの変換機構が存在し、検索サーバから出された DCMES の構造に基づく検索要求をインハウスデータベースに対するクエリに変換し、インハウスデータベースから返される検索結果を DCMES の構造に変換することができれば、この要件は満たされる。

本稿ではふれなかったが、用いるプロトコルとの組み合わせによって、これは様々な実装が可能である。また、ウェブサービス的な実装もあり得ると考えている。

5.　おわりに

本稿では博物館、美術館が公開する文書情報の横断検索に関する検討の一環として、情報記述の構造レベルの相互変換のプロトタイピングとその評価について述べた。

我々の目的は、様々な多様性に対応しながら、博物館情報の横断検索を可能にすることである。このために、レベル I として、記述内容のセマンティクスに関わらない部分でまず記述構造の相互変換を行うことを示した。

今後、レベル II の情報記述内容レベルの相互変換に向けて、更なる検討を重ねていきたい。

参考文献

[1] 博物館情報の知的横断検索のためのフレームワーク：画像電子学会 VMA 研究会、博物館・美術館 DTD-SG, 2002 画像電子学会第 30 回年次大会画像電子ミュージアムテクニカルセッション pp.75-76 (2002).

[2] 博物館情報の知的横断検索の試み：画像電子学会 VMA 研究会、博物館・美術館 DTD-SG, 2002 画像電子学会第 30 回年次大会画像電子ミュージアムテクニカルセッション pp.77-78 (2002).

[3] Dublin Core Metadata Element Set (http://www.dublincore.org/usage/terms/dc/current-elements/)

[4] XSLT (http://www.w3.org/TR/xslt)

[5] Expressing Simple Dublin Core in RDF/XML (http://www.dublincore.org/documents/dcmes-xml/)

[6] Xalan (http://xml.apache.org/xalan-j/)

[7] Xindice (http://xml.apache.org/xindice/)

[8] XML:DB API (http://www.xmldb.org/xapi/)

[9] XPath (http://www.w3.org/TR/Xpath)

[10] XUpdate (http://www.xmldb.org/xupdate)

[11] Xi (http://www.baykit.org/xi/)

[12] BayServer (http://www.baykit.org/bserv/)