

# SVGデータの検索に見るXQueryの可能性

大庭 幹生\*

The Potentiality of XQuery as Found in Some Examples of Querying SVG Data

Mikio Oba\*

XQueryは現在標準化に向けて検討されているXMLに対する汎用的な新しい検索言語である。XMLに対して検索を行うということで、XQueryは従来の検索言語にはない可能性を持っている。本論文では2次元ベクター・グラフィックスを表すSVGデータに対してXQueryを実行する具体例を示し、このXMLに対する検索言語としてのXQueryの持つ可能性について考察する。

XQuery whose standardization is in progress is a new general-purpose language for querying XML data. As it is designed to retrieve data from XML documents, it has some capabilities that none of the existing query languages ever have. In this paper, these are explored by showing some examples to query SVG data, which represent 2-dimensional graphics, by using XQuery.

Key Words & Phrases : XML, XQuery, SVG,

## 1. はじめに

XML( Extensible Markup Language )が1998年に標準化されて以来、データ交換のフォーマットとして、あるいはデータそのものの記述方法として広く利用されてきている。XMLの利用が広まるにつれ、当然XMLをいかに容易かつ効率よく扱うかという点が重要となってくる。そのような中でXMLデータの検索という観点から非常に注目すべき技術が、現在World Wide Web Consortium( W3C )で標準化に向けて検討が重ねられているXML Query Language( XQuery )である。このXQueryに関しては、これまでにWebや雑誌などで様々な技術的な紹介がなされているが[ 1-7 ],その中で検索対象として使用されているXMLデータは、基本的にビジネス・データを模倣した非常に単純なXMLデータである。

筆者はXMLコンソーシアム( <http://www.xmlconsortium.org/> )においてXQueryの技術動向について調査や議論を行い、一般向けにその成果の発表など[ 8, 9 ]を行っている。特に[ 9 ]では、筆者はXQueryの説明、XQueryの処理の解説、SVGに対するクエリ例などを担当した。本論文ではその成果を基に、XQueryの検索対象のXMLデータとして、単なる

ビジネス・データを模倣した単純なXMLデータではなく、2次元ベクター・グラフィックスを表すXMLデータ( SVG )を使用することで、今まで注目されていないXQueryの新しい将来性や可能性が見えてくることを具体的な例を用いて示す。

## 2. XQueryとは

### 2.1 XQueryの概要

XQueryは、W3C[ 10 ] のXML Query Working Groupで標準化が検討されているXMLデータに対する新しい検索言語である。「XQuery 1.0: An XML Query Language」の規格書[ 11 ]を中心として12もの規格書で仕様が検討されている。2001年2月にこれらの規格書の最初のWorking Draftが提出されてから2年半以上経った2003年8月でもまだ全ての文書がWorking Draftのステータスである。

このようにXQueryは非常に巨大な規格であるため、一般的に「XQueryは複雑で難しい」という印象があることは否めない。しかし、現在のWorking Draftを参照する限り、実際にはクエリの記述方法はXMLのツリー構造を考へて直感的に理解しやすい構文となっている。また、IBM、Oracle、Microsoftなどの大手データベース・ベンダーが将来的なサポートを表明しており、Working Draft段階の文書の内容を実装している製品も少しずつ公開されている[ 12 ]ことを見ても、

提出日：2003年8月29日

\*obamikio@jp.ibm.com



```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns:svg="http://www.w3.org/2000/svg"
width="100mm" height="100mm" viewBox="0 0 100 100">
<circle cx="20" cy="20" r="15" fill="#FF0000"
stroke="#000000" stroke-width="0.3"/>
</svg>
```

図2. 円を表すSVGの具体例(ソース)

図2の例では、`<circle>`要素により円を描画することを指定している。円の大きさや色などは、以下の`<circle>`要素で指定している(括弧内は図2の例での設定値)

- cx …… 円の中心のx座標 (20)
- cy …… 円の中心のy座標 (20)
- r …… 円の半径 (15)
- fill …… 円の塗りつぶしの色 (#FF0000 = 赤)
- stroke …… 円の周囲の線の色 (#000000 = 黒)
- stroke-width …… 円の周囲の線の太さ (0.3)

図2のSVGソースをSVG処理プログラムで表示させたものが図3である(座標軸、座標値などは説明のために付け加えたもの。)

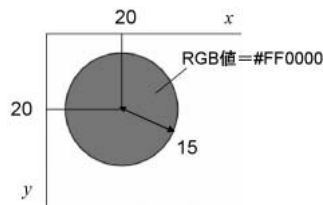


図3. 円を表すSVGの具体例(表示例)

#### 4. SVGに対するXQuery

ここまで、XMLデータに対する新しい検索言語として標準化が検討されているXQueryの内容、およびSVGという2次元ベクター・グラフィックスを表現するXMLデータの説明を行った。ここではSVGデータに対してXQueryを実行することを考え、具体例を示す。

##### (1) 重なり合っている円

検索対象のSVGのソースを図4に、表示結果を図5に示す。

この図形は、半径が5, 10, 15, 20, 25, 30の円を中心の座標をずらして重ねて表示した図形である。この図形に対して、例えば以下のようなクエリを考えることができる。

- (1) 半径が20より小さい円を表示する
- (2) 半径が20より大きい円を表示する

これらのクエリを従来のビットマップ、あるいはベクター・グラフィックスに対して行うことを考えると、非常に困難であるということは容易にわかる。しかし、SVGおよびXQueryであれば非常に簡単に行うことが可能である。上記の(1)と(2)のクエリ例をそれぞれ図6、図7に示す。

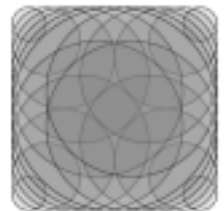


図5. 半径が5, 10, 15, 20, 25, 30の重なり合う円

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg" width="100mm" height="100mm" viewBox="0 0 100 100">
<g>
<circle cx="15" cy="15" r="5" fill="#FF0000" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="15" cy="65" r="5" fill="#FF0000" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="65" cy="65" r="5" fill="#FF0000" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="65" cy="15" r="5" fill="#FF0000" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="20" cy="20" r="10" fill="#FF0033" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="20" cy="60" r="10" fill="#FF0033" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="60" cy="60" r="10" fill="#FF0033" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="60" cy="20" r="10" fill="#FF0033" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="25" cy="25" r="15" fill="#FF0066" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="25" cy="55" r="15" fill="#FF0066" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="55" cy="55" r="15" fill="#FF0066" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="55" cy="25" r="15" fill="#FF0066" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="30" cy="30" r="20" fill="#FF0099" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="30" cy="50" r="20" fill="#FF0099" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="50" cy="50" r="20" fill="#FF0099" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="50" cy="30" r="20" fill="#FF0099" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="35" cy="35" r="25" fill="#FF00CC" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="35" cy="45" r="25" fill="#FF00CC" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="45" cy="45" r="25" fill="#FF00CC" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="45" cy="35" r="25" fill="#FF00CC" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
<circle cx="40" cy="40" r="30" fill="#FF00FF" fill-opacity="0.2" stroke="#000000" stroke-width="0.3"/>
</g>
</svg>
```

図4. SVGデータのソース(circle1.svg)

```
<svg xmlns='http://www.w3.org/2000/svg'
width='100mm' height='100mm' viewBox='0 0 100 100'>
{
  for $c in document("circle1.svg")/svg//circle
  where $c/@r < 20
  return
    <g> { $c } </g>
}</svg>
```

図6. 半径が20よりも小さい円を表示するクエリ

```
<svg xmlns='http://www.w3.org/2000/svg'
width='100mm' height='100mm' viewBox='0 0 100 100'>
{
  for $c in document("circle1.svg")/svg//circle
  where $c/@r > 20
  return
    <g> { $c } </g>
}</svg>
```

図7. 半径が20よりも大きい円を表示するクエリ

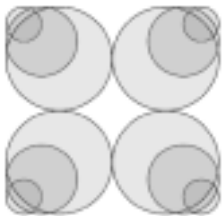


図8. 半径20よりも小さい円のみを表示

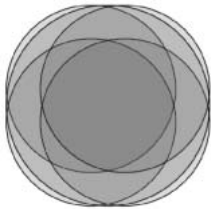


図9. 半径20よりも大きい円のみを表示

図6, 図7を見てわかるように, クエリ自体は非常に単純である. 実際にこれらを図5のSVGソースに対して実行した結果を表示すると, それぞれ図8, 図9に示す通りとなる.

上記では半径の大きさという一つの属性のみを検索対象としているが, 当然検索SVGのソースに記述されている属性であればどのような属性に対しても検索が可能である. このように考えると, XMLベースのSVGで図形が記述されていることと, そのXMLに対する汎用検索言語であるXQueryを使用することで,

従来では全く考えられなかった柔軟性を持った新しい検索が可能となることがわかる.

## (2) 市松模様の変形

次に, XQueryによりオリジナルのデータから変形させた結果を得ることができるという例を示す. 検索対象のSVGの表示結果を図10に示す(ソースは省略)

この図形は縦10, 横10のサイズの正方形が, 4つ×4つに並んで, 市松模様(チェッカーボード)に色分けされている図形である. この図形に対して横方向(x軸方向)の長さを2倍に拡大した図形を

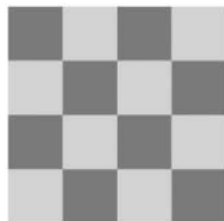


図10. 市松模様

```
<svg xmlns='http://www.w3.org/2000/svg'
width='80mm' height='40mm' viewBox='0 0 80 40'>
{
  for $r in document("rect1.svg")/svg//rect
  return
    <g>
      <rect x={$r/@x*2} y={$r/@y}
width={$r/@width*2} height={$r/@height}
fill={$r/@fill}; stroke="none"/>
    </g>
}</svg>
```

図11. x軸方向に2倍に拡大するクエリ例

戻すようなクエリを記述することが可能である. このクエリ例を図11に示す.

実際にこれを図11のSVGソースに対して実行した結果の表示結果を図12に示す.

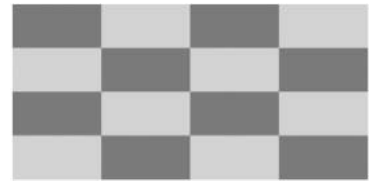


図12. 市松模様に対するクエリ結果

検索結果として新しいXMLデータを作成することが可能であることはXQueryの大きな特長の一つである. このため, 上記のように検索対象のSVGデータを基にして別の新しいSVGデータを作り出すということも可能である.

## 5. SVGデータに対する

### XQueryを実行することに関する考察

前章で見たように, XMLで記述されているSVGデータに対してXQueryで検索を行うことは容易である. ただし, そこには単にXMLデータに対してXQueryで検索を行うということ以上の意味があると筆者は考えている.

XMLは, データ自体が意味を持つことが可能である. 例えば, 図4の文字列は, その文字列自体が持つ意味とは別に, 図5で表現される図形であるという意味も持っている. 従って, XMLに対して検索を行うということは, 単にそのXMLの文字列を検索していること以上に, そのXMLが表現しているもの自体を検索しているとも解釈できる. 例えば, 図6のクエリは, 単にXMLデータの中のcircleという要素で, rという属性の値が20より小さい要素を取り出しているだけではなく, 図5で表現される図形(複数の円)の中から半径が20より小さい円を取り出しているとも読むことができる.

このように考えると, XQueryは図形に対して検索を実行することができるということがわかる. この意義は他の検索言語, 例えばリレーショナル・データベースにおいて表に対して実行するSQL(Structured Query Language)と比較することでより明確になる. SQLが

検索対象とするデータは、そのアプリケーションにとっては何らかの意味を表しており、検索結果も何らかの意味を表しているが、その意味はSQLを実行したアプリケーション以外には理解することは不可能である。従って、取得した結果を意味のあるものにするためには、意味を理解しているアプリケーションが全ての処理を行う必要がある。しかし、XQueryの場合は検索対象のデータの意味はXQueryを実行するアプリケーション以外でも理解することが可能であり、XQueryの結果も同様である。これは上記のSVGの例の表示(図5,8,9,10,12)はXQueryアプリケーションが独自に表示しているわけではなく、一般的なSVG処理プログラムが表示していることからわかる。

このように考えると、XQueryは非常に大きな可能性を持っていることがわかる。なぜなら、XMLにより表現されるデータは当然SVGだけではなく、マルチメディア・データ、ビジネス・データなど様々な業界において非常に多く(理論的には無限に)存在するからである。それらのデータを他のデータに変換することなく、そのままXQueryで検索することが可能となる。

別の例としては、JavaオブジェクトとXMLとのマッピングを行うことで、Javaオブジェクトを直接XQueryで検索することが挙げられる。JAXB[18]やRelaxer[19]は、XMLデータを扱うJavaクラスを自動生成し、Javaから容易にXMLデータを扱えるようにする仕組みであり、この仕組みではJavaオブジェクトをXMLデータに等価かつ容易にマッピングすることができる。現在はJavaオブジェクトを保管するには、例えばリレーショナル・データベースの表の列にマッピングして保管するという手間をかけていることが多いが、その代わりにJavaオブジェクトをXMLデータとして保管し、検索をXQueryを使用して実行することで、より単純でわかりやすいオブジェクトの保管および検索が実現できる。これも、つまりJavaオブジェクトをXMLで表記することによって、汎用的な検索言語であるXQueryを使用することが可能となる。

以上の通り、検索対象がXMLであるということにより従来であれば検索対象とみなされていなかったデータを直接検索言語により検索することができる。ここにXQueryの新しい可能性があり、潜在的な大きなビジネス・チャンスがあると筆者は考えている。

## おわりに

本論文では、XQueryをグラフィックス・データを表すXMLデータであるSVGに対して実行する例を通じて、従来であれば検索を行う対象ではなかったデータに対しても検索が実行可能となるというXQueryの可能性について指摘した。

今後XQueryの標準化と共に実装するソフトウェア製品は多くなり、XQueryを新たに学び始める技術者の数が増加することが予想される。本論文が、その技術者の理解の助けとなり、新たなビジネスを生むきっかけとなれば幸いである。

## 謝辞

本論文は、2003年度XMLコンソーシアムXMLテクノロジー部会XML-DBワーキング・グループでの活動を基にしている。活動にあたり、山本浩一氏(NTTソフトウェア株式会社)、千葉恭弘氏(株式会社電通国際情報サービス)、根来元氏(日本ユニシス株式会社)らメンバーの方々と有意義な議論を行った。ここに記して謝意を表す。

## 参考文献

- [1] Don, Chamberlin, *XQuery: An XML query language*, IBM Systems Journal, Vol41, No.4, pp597-615, 2002
- [2] 戌亥稔, 標準化目前: 注目のXML問い合わせ言語 XQuery, @IT, 2002/02/09, <http://www.atmarkit.co.jp/fxml/tanpatsu/14xquery/xquery01.html>
- [3] 戌亥稔, XQueryチュートリアル(1)~ XQueryを実体験してみる, @IT, 2002/08/07, <http://www.atmarkit.co.jp/fxml/tanpatsu/19quip/quip01.html>
- [4] Dan Maharry, Rogerio Saran, Kurt Cagle, Mark Fussell, Nalleli Lopez, *Early Adopter XQuery*, Wrox Press, 2002
- [5] Process XML using XQuery, IBM developerWorks, [http://www6.software.ibm.com/developerworks/education/x-xquery/\(2003/08\)](http://www6.software.ibm.com/developerworks/education/x-xquery/(2003/08))
- [6] Howard Katz, An Introduction to XQuery, IBM developerWorks, 2001/06 [http://www-6.ibm.com/jp/developerworks/xml/011116/j\\_x-xquery.html](http://www-6.ibm.com/jp/developerworks/xml/011116/j_x-xquery.html)
- [7] Kevin Williams, XML for Data: An early look at XQuery, 2002/02/01 <http://www-106.ibm.com/developerworks/xml/library/x-idxqry.html?dwzone=xml>
- [8] 山本浩一, 大庭幹生, XQueryってどんなもの?, 第3回XMLコンソーシアムDay, 2002/11/07, <http://www.xmlconsortium.org/>
- [9] 山本浩一, 千葉恭弘, 大庭幹生, ネイティブXML-DBに格納したXML(SVGデータ)をXQueryで検索するツールのデモ, 第2回XMLコンソーシアムWeek, 2003/05/28, <http://www.xmlconsortium.org/>

- [ 10 ] W3C (World Wide Web Consortium),  
<http://www.w3.org/> (2003/08)
- [ 11 ] XQuery 1.0: An XML Query Language W3C Working  
Draft 22 August 2003,  
<http://www.w3.org/TR/xquery/>
- [ 12 ] W3C Architecture Domain XML Query (XQuery) ,  
<http://www.w3.org/XML/Query#products> (2003/08)
- [ 13 ] XML Path Language (XPath) 2.0 W3C Working  
Draft 22 August 2003,  
<http://www.w3.org/TR/xpath20/>
- [ 14 ] XQuery 1.0 and XPath 2.0 Functions and Operators  
W3C Working Draft 02 May 2003,  
<http://www.w3.org/TR/xpath-functions/>
- [ 15 ] XML Query Use Cases W3C Working Draft  
22 August 2003,  
<http://www.w3.org/TR/xquery-use-cases/>
- [ 16 ] Scalable Vector Graphics (SVG) 1.1 Specification  
W3C Recommendation 14 January 2003,  
<http://www.w3.org/TR/SVG11/>
- [ 17 ] Scalable Vector Graphics (SVG) 1.2 Specification  
W3C Working Draft 15 July 2003,  
<http://www.w3.org/TR/SVG12/>
- [ 18 ] Java Architecture for XML Binding (JAXB),  
<http://java.sun.com/xml/jaxb/> (2003/08)
- [ 19 ] Relaxer Version 1.0 RC2,  
<http://www.relaxer.org/> (2003/08)

.....



日本アイ・ピー・エム  
 システムズ・エンジニアリング株式会社  
 ITスペシャリスト  
**大庭 幹生** Mikio Oba

**[ プロフィール ]**  
 1999年 ,日本アイ・ピー・エム システムズ・エンジニアリング入社 .  
 リレーショナル・データベース製品であるDB2 Universal Database  
 for Linux, UNIX and Windows の技術サポートを担当 .多くのお客様  
 のシステム構築プロジェクトに対する技術支援 ,新バージョンの機能  
 検証や研修の企画・実施 ,解説資料の執筆などを行っている .