Kyoto University, Institute for Research in Humanities

ＣＯＥサマーセミナー2006年：XML文書検索

Christian Wittern

2006-09-13

1. はじめに

プレーン・テキストと違って、XML文書は階層をもっているデータ形式である。こうした文書のなかで、文書の構造を意識して検索すると、より充実した結果が得られる。ここではこうした検索方法を紹介することにする。

そのためにまずXMLそのものを理解しなければならない。それからXML処理のために基本的なツールを使って、XML文書の中身をのぞく。ここで使うツールは標準化団体W3C(World Wide Web Consortium)が開発してきたXpathとその拡張であるXQuery[1]となる。

2. XMLとは？

2.1. 歴史

1960　年代から、記述的マークアップ方法の標準化を目指した、多くのプロジェクトがあったが、中でもIBMのCharles Goldfarb率いるメンバーが開発したGML (Generalized Markup Language) が最も成功し、さらに、ANSI (American National Standards Institute) とともにSGML (Standard Generalized Markup Language) を開発し、1986 年にISO (International Standards Organizations) はSGML を標準(ISO 8879: Information Processing Text and Office Systems -Standard Generalized Markup Language)として認可した。SGMLはマークアップ言語というよりメタ言語（言語を記述するための言語、ここではマークアップ言語を定義する言語）の側面のほうがむしろ強く、テキストに付加された情報と本文とを区別するのに使用される文字列のようなものをいう。SGML は特定用途向けマークアップ言語を定義するのに使用でき、これらをSGMLのアプリケーション（応用）と呼ばれる。そのようなアプリケーションは、文書のどの場所にどんなタグが使われるかといった仕様書を作成することによって構成される。

[1] *XQuery 1.0: An XML Query Language W3C Candidate Recommendation 8 June 2006*, http://www.w3.org/TR/xquery/。 2006年9月現在、W3Cで標準化作業が進展中であり、勧告(Recommendation)一歩手前の勧告候補(Candidate Recommendation)となっている。

DTD (Document Type Definition) と呼ばれる仕様書は書いた文書が規格に沿って書かれているかをチェックするのに利用でき、もし仕様に沿ってなければすぐに判るようになっている。こういう目的に使われるソフトウェアをパーサ（構文解析ツール）と呼び、文書の検証過程に使われる。

2.2. HTML

SGML の一つの応用にHTML (Hyper Text Markup Language) がある。これはWWW (World Wide Web) の基本的なマークアップ言語である。HTML はウェブページを作成する際に良く使われる要素（タグ）のセットを定義している。例えば、段落なら<p>、レベルの異なる見出しに対しては <h1>、<h2> などがあり、ページ内に画像を埋め込むには<img>、他のページへのリンクを張るには<a> といったタグが用意されている。非常に限られたセットではあるが、これに対応するソフトウェアを書くことは簡単であり（この手のソフトウェアはブラウザまたはウェブブラウザと呼ばれている）、WWW の大規模な成長と継続的なサポートを見れば、十分に成功しているといえる。

しかしながら、HTMLで有効なタグセットは少なかった、そしてすぐに限界が見えてしまい、各ブラウザメーカーは競って独自のタグを考案しようとした、当然独自のタグは他のブラウザには対応していないだろう。他にはSGML の仕様が複雑であり、検証ソフトを作ることが困難という問題もすぐに生じた。そのようなことから、多くのユーザーは使用したブラウザが文書を表示することができるなら、画面を見ることでHTML 文書の検証の代わりにしていた。一方で、ブラウザメーカはユーザの底辺を拡大するために、多少文法が間違っていてもできるだけ推測して画面に表示するように努力した。

2.3. SGML からXML へ

この状況を解決して、最適な道筋を経てWWW に提供するために、W3C は使い易く、実装し易いSGML の簡易型のバージョンを開発する任務を主だった専門家のワークグループに課した。W3C とはWWW で利用される技術の標準化をすすめる団体のことで、WWW 技術に関わりの深い企業、大学・研究所、個人などが集まって、1994 年10 月に発足した。SGML の簡易バージョンの開発の結果、1998 年2 月にXML (eXtensible Markup Language) としてW3C 勧告として公表された。

XML は文書型（ドキュメントタイプ）を簡単に定義でき、また、定義なしでも動作する。XML は多言語文書のために文字コードにUnicode を採用していて、しかもXML 文書の検証に対して簡潔であり、効率的な規則がある。産業界はXMLの開発と標準化に関わり、即座に成功を

立証した。現在では、XMLはテキストエンコーディングに広く使用されるだけではなく、メタデータ、データ交換、およびメッセージング（データの送受信）に使用されたりする。XMLマークアップのボキャブラリには、コンピュータグラフィック、数式、化学業界、地理情報、書誌学の記録、企業取引、報道などといった多くの分野に適した仕様がある。

2.4. XML を詳しく見ていこう

XML 文書は7 つの異なった構成要素を含んでいる：

- 要素(Element)
- テキスト(Text)
- 属性(Attribute)
- 実体(Entity)
- コメント(Comment)
- 処理命令(Processing Instructions)
- CDATA （文字データ(Character Data)）

さらに、文字コードやXML 文書であると認識させるためや、DTD (DOCTYPE 宣言) を指し示すために文書に先だつ幾つかの構成要素がある。 最小構成のXML 文章は以下のようになる：

```
<firstDoc n="1">
<p>This is an instance of a "firstDoc" document</p>
</firstDoc>
```

今までのところ述べてこなかったXMLファイルを規定するいくつかのルールがある。一つは、最初の要素が他のすべての要素を含まなければならない、これをルート要素という。上の例では<firstDoc> がルート要素になる。このケースでは、ルート要素は<p> という要素を含んでいる。もう一つは、すべての要素が互いにネストになっていなければならない。その結果、下記の例のような構造は妥当なXML ではない：

```
<p>Two <a>elements, <b>overlapping</a> each</b> other.</p>
```

要素<a> の開始の後に、要素<b> が開始しているが、要素<b> が終了する前に要素<a> が終了しているために妥当ではない。妥当なXML にするには以下のように要素<b> を終了してから、要素<a> を終了させなければならない：

```
<p>Two <a>elements, <b>overlapping</b> each</a> other.</p>
```

例では、要素<firstDoc> は属性名「n」、属性値「1」を含んでいる。通常、属性は要素に含まれたテキスト（要素の値）に関する追加情報を伝え、また、属性の指定方法を規定するいくつかのルールがある。最も重要なルー ルは属性の値の開始と終了はシングル・クォーテション「'」かダブル・クォーテション「"」のどちらかのペアで区切らなければならず、属性の名前と属性の値とは「=」で繋げなければならない。 最初のうちは分かりにくいと思うが、専用のXML エディタなら細かなことに注意を払ってくれるので、誰で もすぐに使いこなせるようになる。

2.5. XML スキーマ言語

前節で述べたように、オリジナルのXMLの仕様ではDTDを使った文書構造とコンテンツの設計（いわゆるスキーマ）を可能にしていた。どのように使用するのかを上記の例を使い、XML文書にDTD の定義を含めると以下のようになる：

```
<!DOCTYPE firstDoc [
<!ELEMENT firstDoc (p+) >
<!ATTLIST firstDoc n CDATA #IMPLIED>
<!ELEMENT p (#PCDATA)>
]>
<firstDoc n="1">
<p>This is an instance of a "firstDoc" document</p>
</firstDoc>
</firstDoc>
```

1 行目の<!DOCTYPE [は「firstDoc 」というDTD があることを宣言し、2 行目は要素<firstDoc> が<p> 要素を1 つ以上持たなけれ(firstDoc)ばならない。3 行目で要素<firstDoc> が任意で属性「n」を持つことを意味し ている。DTD の詳細は「XML 入門」を参照。 こうしている間にも、他のスキーマ言語が開発されている。今のところ、XML 文書では以下の3 種類が広く使 われている：

- DTD はSGML の開発ともにに開発された。ファイル拡張子は「.dtd」
- XML Schema はW3C によって開発された。ファイル拡張子は「.xsd」
- Relax NG （リラクシングと発音）はJames Clark とMurata Makoto （村田真）によって開発され、OASIS とISO で承認されている。ファイル拡張子は「.rng または.rnc」

ここでは細かな違いは述べないが、当面は幾つかの言語があることを知っていれば十分である。それら全ては、文書の形式を定義し、文章が妥当であるかを検証するのに使われるが、それぞれ使用制限は異なる。通常どのタイプのスキーマで書かれているかは、ファイルの拡張子から推測できる。例えば、図1 で示すように、XML エディタの<oXygen/> は与えられたファイルを検証するためにスキーマと関連付ける機能を持っている。より詳しい<oXygen/> の使い方はユーザガイドを見て欲しい。

3. XPath

XML Path Language (XPath; XMLパス言語) は、マークアップ言語 XML に準拠した文書の特定の部分を指定する言語構文である。 XPath自体は簡潔な構文(式言語)であり、XMLに準拠したマークアップ言語ではない。 W3Cで開発され、1999年11月16日に XML Path Language (XPath) 1.0 が XSL Transformations (XSLT) 1.0 と同時に勧告として公表された。 XPathは、XSLT と XSL-FO とともにスタイルシート技術XSLの構成要素と位置づけられている。 2006年9月現在、W3C でXPath 1.0 の次期バージョンの制定作業が進められており、XPath 2.0 が勧告候補となっている。 ここではXPath 2.0を使用する。[2]

3.1. XPath 構文

XPathで最も一般的な式は、ロケーションパスである。 ロケーションパスにより、XML文書のあるノード(現在のコンテクストノード)を基準として、ノード集合（0 以上のノード）が指定される。

ロケーションパスを構成する各ロケーションステップは、次の3つの要素から構成される。

- 軸 (axis)
- ノードテスト (node test)
- 述語 (predicate)

XPathの例：

```
//div[@type='poem']
/TEI/text/body/p/rm/@key]
```

[2] ここではXPathの細かい説明を省略し、フリー百科事典『ウィキペディア（Wikipedia）』を参照する。
http://ja.wikipedia.org/wiki/XML_Path_Language

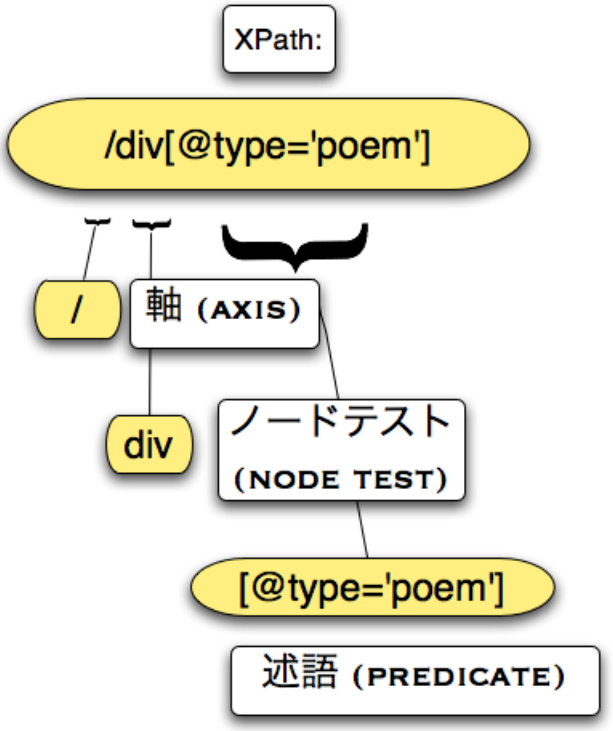

図1.

この三つの要素で複雑なXPathを組み合わせる：

```
/div[@type='poem']/lg[@type='近體詩:七言:七言排律']/l[@xml:lang="zh-Latn-x-pinyin"]
```

述語として、位置の順序も使える：

```
/七言絕句[1]/絶句[1]/句[2]
```

述語の中或いはノード集合の各ノードに対してはXPath関数でさらに条件を付らけれる或いは処理できる：

```
/七言絕句[1]/絶句[1]/句[contains(., '寒山')]
```

```
/七言絕句[1]/絶句[1]/句/substring(., string-length(.))
```

3.2. XML文書におけるXpath軸

Xpath軸は文書の中の一つのノードから周辺にあるノードを指す。結果はノード集合である。以下はこのXML文書で要素をたどる１１個の軸の使用を確認して欲しい。全部で１３個の軸があるが、属性軸(attribute axis)と名前空間軸(namespace axis) は要素内の軸となる、そのあとを紹介する。

```
<七言絶句>
  <作者>張繼</作者>
  <題名>楓橋夜泊</題名>
  <絶句>
    <句 n="1">月落烏啼霜滿天，</句>
    <句 n="2">江楓漁火對愁眠。</句>
    <句 n="3">姑蘇城外寒山寺，</句>
    <句 n="4">夜半鐘聲到客船。</句>
  </絶句>
</七言絶句>
```

3.2.1. 要素をたどる軸

3.2.1.1. ancestor

コンテクストノードの祖先ノード

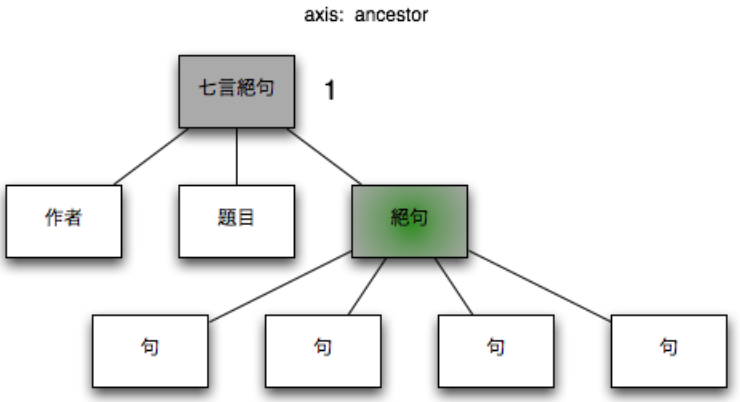

図 2.

XPath	$node/ancestor::*
コンテクストノード	/七言絶句[1]/絶句[1]
コンテクストノード + XPath	/七言絶句[1]/絶句[1]/ancestor::*

3.2.1.2. ancestor-or-self

コンテクストノード自身とコンテクストノードの祖先ノード

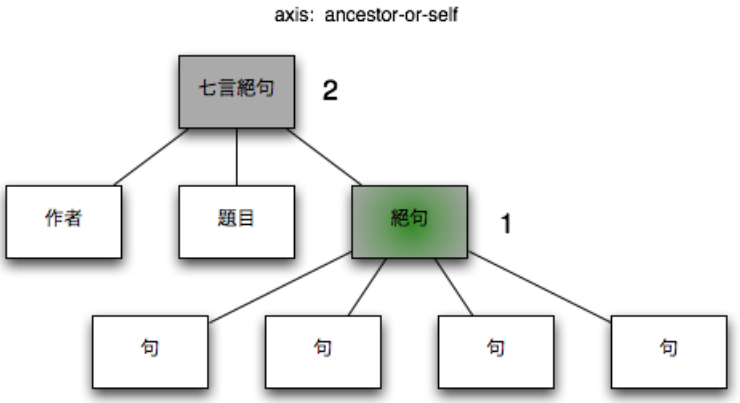

図 3.

XPath	$node/ancestor-or-self::*
コンテクストノード	/七言絶句[1]/絶句[1]
コンテクストノード + XPath	/七言絶句[1]/絶句[1]/ancestor-or-self::*

3.2.1.3. child

コンテクストノードの子ノード

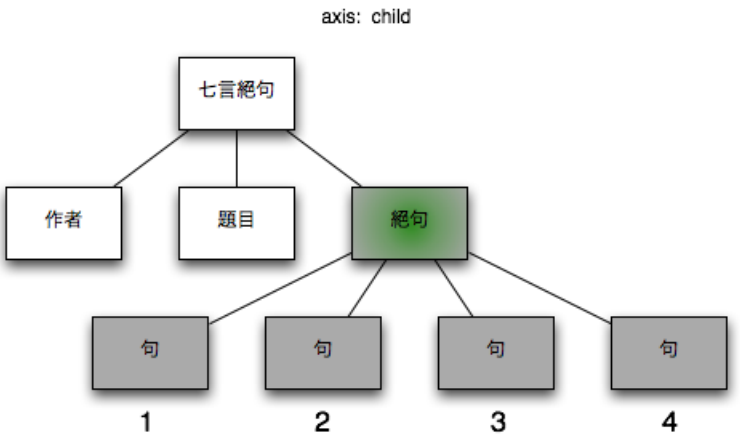

図 4.

XPath	$node/child::*
簡潔構文XPath	$node/*
コンテクストノード	/七言絶句[1]/絶句[1]
コンテクストノード + XPath	/七言絶句[1]/絶句[1]/child::*

3.2.1.4. descendant

コンテクストノードの子孫ノード

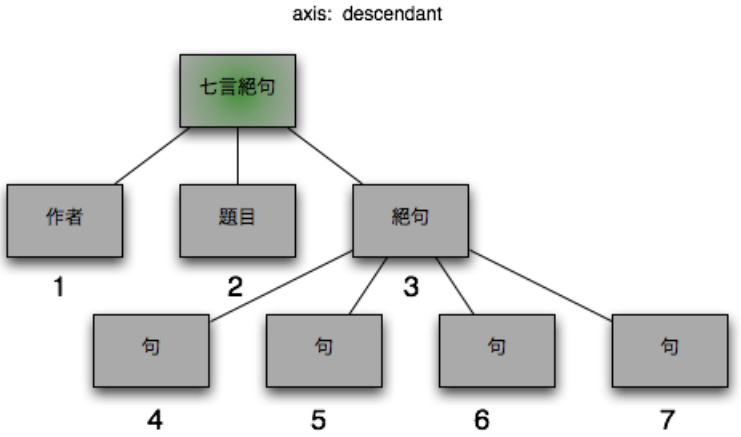

図5.

XPath	$node/descendant::*
コンテクストノード	/七言絶句[1]
コンテクストノード + XPath	/七言絶句[1]/descendant::*

3.2.1.5. descendant-or-self

コンテクストノード自身とコンテクストノードの子孫ノード

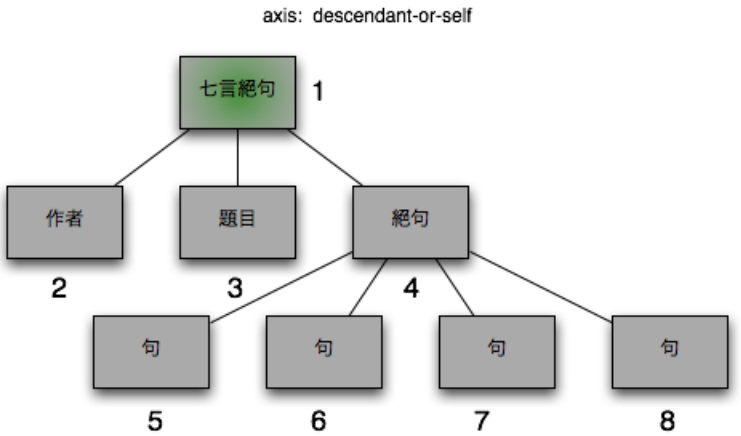

図6.

XPath	$node/descendant-or-self::*
コンテクストノード	/七言絶句[1]

コンテクストノード + XPath　　/七言絶句[1]/descendant-or-self::*

3.2.1.6. following

XML文書の文書順でコンテクストノードより後方にある全てのノード

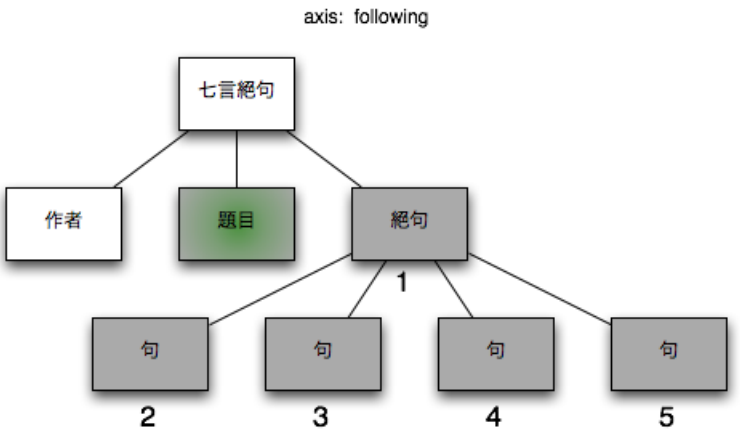

図 7.

XPath　　$node/following::*

コンテクストノード　　/七言絶句[1]/題名[1]

コンテクストノード + XPath　　/七言絶句[1]/題名[1]/following::*

3.2.1.7. following-sibling

コンテクストノードの兄弟ノードのうち後方のノード

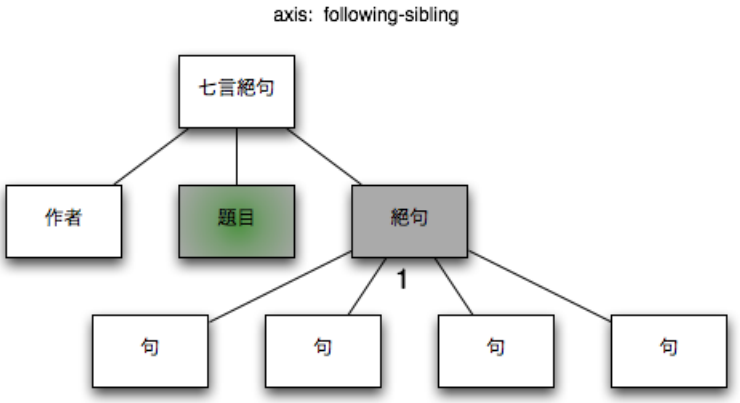

図 8.

XPath　　$node/following-sibling::*

コンテクストノード　　/七言絶句[1]/題名[1]

コンテクストノード + XPath　/七言絶句[1]/題名[1]/following-sibling::*

3.2.1.8. parent

コンテクストノードの親ノード

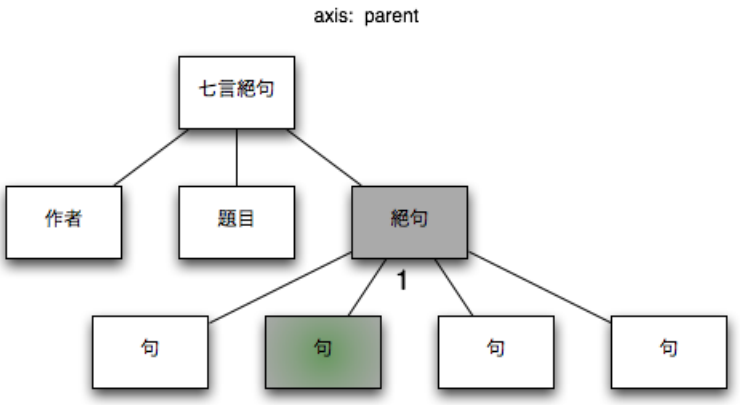

図 9.

XPath　　$node/parent::*

コンテクストノード　　/七言絶句[1]/絶句[1]/句[2]

コンテクストノード + XPath　　/七言絶句[1]/絶句[1]/句[2]/parent::*

3.2.1.9. preceding

XML文書の文書順でコンテクストノードより前方にある全てのノード

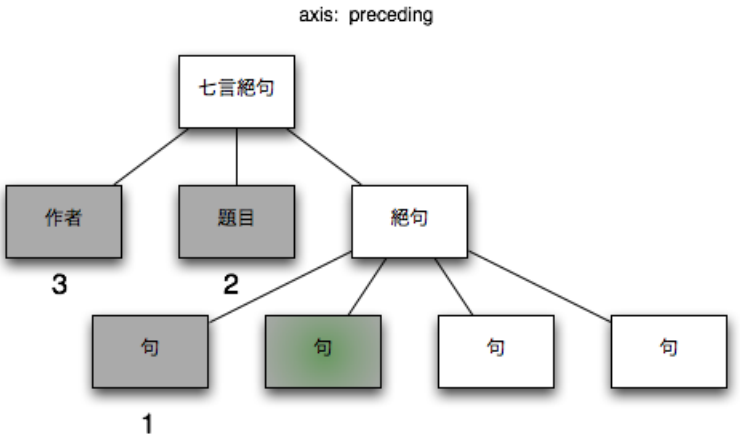

図 10.

XPath	$node/preceding::*
コンテクストノード	/七言絶句[1]/絶句[1]/句[2]
コンテクストノード + XPath	/七言絶句[1]/絶句[1]/句[2]/preceding::*

3.2.1.10. preceding-sibling

コンテクストノードの兄弟ノードのうち前方のノード

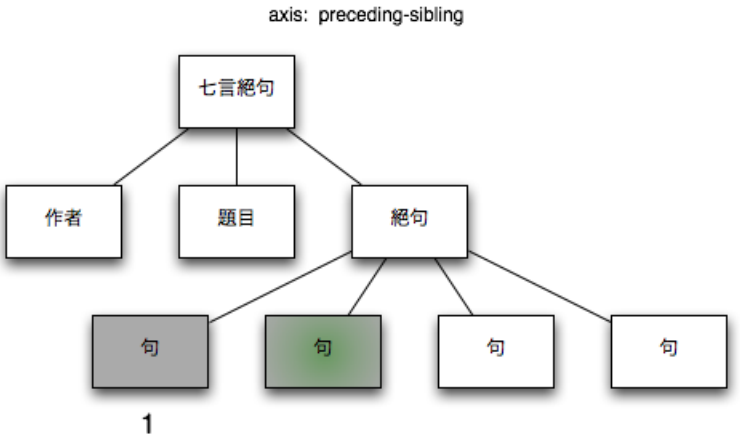

図 11.

XPath	$node/preceding-sibling::*
コンテクストノード	/七言絶句[1]/絶句[1]/句[2]
コンテクストノード + XPath	/七言絶句[1]/絶句[1]/句[2]/preceding-sibling::*

3.2.1.11. self

コンテクストノード自身

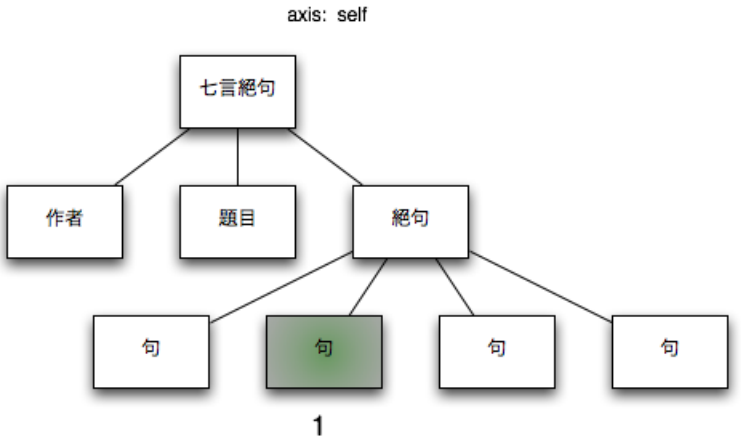

図 12.

XPath	$node/self::*
コンテクストノード	/七言絶句[1]/絶句[1]/句[2]
コンテクストノード + XPath	/七言絶句[1]/絶句[1]/句[2]/self::*

3.2.2. 要素内軸

3.2.2.1. attribute

属性ノードを選択する

XPath	$node/attribute::n
簡潔構文XPath	$node/@n
コンテクストノード	/七言絶句[1]/絶句[1]/句[2]
コンテクストノード + XPath	/七言絶句[1]/絶句[1]/句[2]/attribute::n

3.2.2.2. namespace

名前空間ノードを選択する

XPath	$node/namespace::*
コンテクストノード	(//tei:p[1])[1]
コンテクストノード + XPath	(//tei:p[1])[1]/namespace::node()

3.3. XPath 関数

XPath 2.0に１００個以上の関数が定義されているが、XQueryはこれを更に拡張し、XQuery用のXPath関数を定義する。ここではそのなかから一部分だけを紹介する。

3.3.1. doc()

入力となるXML文書を開く:

定義	doc(*ファイル名*)
返還値	XML文書（ルートノード）
例	doc('bjy.xml')

3.3.2. contains()

コンテクストノードに指定した文字列が含めていることを調べる（コンテクストノードはXPathで '.' 表示する）：

定義	contains(*文字列, 検索語*)
返還値	ブール値(true, false)
例	//句/contains(., '寒山')

述語にも使える：

```
//句[contains(., '寒山')]
```

3.3.3. starts-with(), ends-with()

文字列の頭(starts-with())或いは末（ends-with()）に指定した文字列が含まれていることをしらべる：

定義	starts-with(*文字列, 検索語*)
返還値	ブール値(true, false)
例	//句[starts-with(., '月落')]
例	//句[ends-with(., '寒山寺')]

3.3.4. substring()

文字列の一部を取り出す、例えば第一字だけをとりだす：

定義	substring(*文字列*, n, m)

n	取り出す文字列の第一字
m	取り出す文字列の字数
返還値	文字列
例	//句/substring(., 1, 1)

3.3.5. substring-after(), substring-before()

ある検索語の後(substring-after)或いは前(substring-before)の部分を文字列から取り出す：

定義	substring-after(*文字列*,*検索語*)
返還値	文字列
例	substring-after('天寶-1', '-')
例	substring-before('天寶-1', '-')

3.3.6. string-length()

引数として渡す文字列の長さ (文字の数) を返す。

定義	string-length(*文字列*)
返還値	数値
例	//句/substring(., string-length(.), 1)

3.3.7. count()

引数のノード集合のノードの数を返す。

定義	count(*ノード集合*)
返還値	数値
例	//句/substring(., string-length(.), 1)

3.3.8. distinct-values()

引数のノード集合に対して複数の値を取り去って返す。

定義	distinct-values(*ノード集合*)
返還値	値の集合
例	distinct-values((1, 1, 3, 1, 4, 2, 3))

3.3.9. xs:int()

引数の文字列を数字として返す。'xs:'はXML Schemaに定義されている名前空間識別子、XQueryのなかにもすでに定義されている。

定義	xs:int(*文字列*)
返還値	数字
例	xs:int('010')

3.3.10. string()

引数のノードを文字列としての値を返す。

定義	string(*ノード*)
返還値	文字列
例	string(//tei:p[1])

3.3.11. name()

引数のノードの名前(node-name)をて返す。

定義	name(*ノード*)
返還値	文字列
例	//tei:dm/parent::tei:*/name()

4. XQuery

XQueryはXML問い合わせ言語のひとつである。 XQueryはXMLの自由構造型データに対して、関係構造型データであるRDB(Relational Database)におけるSQLと同等の機能を持つデータ検索インタフェースを与えようというもの。そのサブセットであるXPathの拡張やFLOWR(FOR-LET-WHERE-ORDER-RETURN)式が実現されたことで、SQLにおけるクエリのネストやテーブルのJOINに相当する、複雑な検索やXMLドキュメントの結合を行うことができる。

XQueryはSQLのSelect節、From節、Where節、Order節などのようなものをFLWOR表現式のfor、let、where、order、return句などで実現して複数のXML文書にたいして選択、射影、直積、結合などを実行することが出来る。

表 25.

for	選択
let	射影
where	直積
order by	並べ替え
return	結合

以上のキーワードはXQueryの基本となるが、すべて必要ではない。

4.1. 優しいXQuery実例

演習資料にxquery1.xqというファイルのなかの一番簡単なXQueryの例がある：

```
xquery1.xq:
//句
```

実はこれはただのXPathで、XQueryの機能はまだ使われていない。同じものをFLWOR式で書くとこうなる(xquery2.xq)：

```
xquery2.xq:
for $k in //句
return
$k
```

「ノード集合'//句'のなかの各ノードを順番でだして、$kの値として処理する。ここでの処理は$kの値を返す。」ということが起こる。もちろん例えばこうした処理も可能になる：

```
for $k in //句
return
substring($k, 1, 1)
```

4.2. 名前空間とXQuery

xquery3.xqで違うファイルの内容を処理する。それはTEIのXML文書（P5）であり、なかにはXML名前空間が定義されて、XQueryでも同じ名前空間でなければ処理できない。xquery3.xq一行目にまず名前空間宣言があり、tei:という名前空間識別子をTEIの名前空間("http://www.tei-c.org/ns/1.0")にマップする。このXQueryで以下のTEIの名前空間をtei:で省略可能となる。

```
xquery3.xq:
declare namespace tei="http://www.tei-c.org/ns/1.0";

for $l in doc('bjy.xml')//tei:l[contains(., '心')]
let $pin := $l/following-sibling::tei:l[1]
return
<l pin="{$pin}">{string($l)}</l>
```

xquery3.xqにも初めて"let"でFLWOR式内の変数を定義する（第４行）。

4.3. XQuery式を出力XML内に書き込む

XQueryの結果を出力するときには、XML文書のツリーのなかに書き込むことがよくある：

```
xquery4.xq:
declare namespace tei="http://www.tei-c.org/ns/1.0";
let $q := '心'
return
<div xmlns="http://www.tei-c.org/ns/1.0">
{
for $l in doc('bjy.xml')//tei:l[contains(., $q)]
let $pin := $l/following-sibling::tei:l[1]
let $title := $l/ancestor::div[1]/head[1]
return
<p>
<head>{string($title)}</head>
<l pin="{$pin}">{string($l)}</l>
</p>
}
</div>
```

4.4. HTMLを出力する

```
xquery5.xq
declare namespace tei="http://www.tei-c.org/ns/1.0";


<html>
<head><title>検索結果</title></head>
<body>
<ul>
{
for $r in doc('zztj-sample.xml')//tei:rm[@key='r00793']
let $h := $r/ancestor::tei:div[2]/tei:head[1]/@n
let $y := xs:int(substring-after($h, '-'))
order by $y
return
<li>{string($h)}:{$r}</li>
}
</ul>
</body>
</html>
```