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

XML コンソーシアム XMLテクノロジー部会
XML-DB WG

NTTソフトウェア(株)
山本 浩一（ymmt@po.ntts.co.jp）

（株)電通国際情報サービス
千葉 恭弘（chiba@isid.co.jp）

日本アイ・ビー・エム システムズ・エンジニアリング（株）
大庭 幹生（obamikio@jp.ibm.com）

XQueryの登場の背景

- リレーショナルデータベースのクエリ言語
  - リレーショナルデータベースには標準のクエリ言語であるSQLがある。

リレーショナルデータベースは、どの製品であっても、同じSQLという標準のクエリ言語を使用して、データを検索することが出来ます。
そのため、リレーショナルデータベースを使いたい人は、SQLを覚えさえすれば、どんなリレーショナルデータベース製品でも使うことが出来るようになります。
このことが、リレーショナルデータベースを世の中に広く普及させる原動力になったといえるでしょう。

XQueryの登場の背景

- リレーショナルデータベースのクエリ言語
  - リレーショナルデータベースには標準のクエリ言語であるSQLがある。

XQueryの登場の背景

- XMLネイティブデータベースのクエリ言語
  - XMLネイティブデータベースに対する標準のクエリ言語としてはXPathがある。

XMLネイティブデータベースの世界では、XPathが標準のクエリ言語として受け入れられています。
XMLを格納できるデータベースは、ほとんどの製品がこのXPathによる検索機能をサポートしています。
しかし、XPathの機能だけでは、XMLからある特定の条件を満たしたノードを取り出すことはできますが、取り出したノードを組み合わせて新しいXMLのデータ構造を作ることはできません。

XQueryの登場の背景

- XMLネイティブデータベースのクエリ言語
  - XMLネイティブデータベースに対する標準のクエリ言語としてはXPathがある。

例えば、XPathを用いて、

```
/students/student[@name="山本"]
```

これから

```
<students>
 <student name="山本">
    <teacher>大庭</teacher>
 </student>
 <student name="千葉">
    <teacher>大庭</teacher>
 </student>
<students>
```

これ

```
<student name="山本">
   <teacher>大庭</teacher>
</student>
```

を取り出すことはできますが、

XQueryの登場の背景

- XMLネイティブデータベースのクエリ言語
  - XMLネイティブデータベースに対する標準のクエリ言語としてはXPathがある。

これから

```
<students>
 <student name="山本">
   <teacher>大庭</teacher>
 </student>
 <student name="千葉">
   <teacher>大庭</teacher>
 </student>
<students>
```

これ

```
<teacher name="大庭">
   <student>山本</student>
   <student>千葉</student>
</teacher>
```

を取り出すような、取り出したノードを組み合わせて新しいXMLのデータ構造を作るような検索を行うことは出来ません。

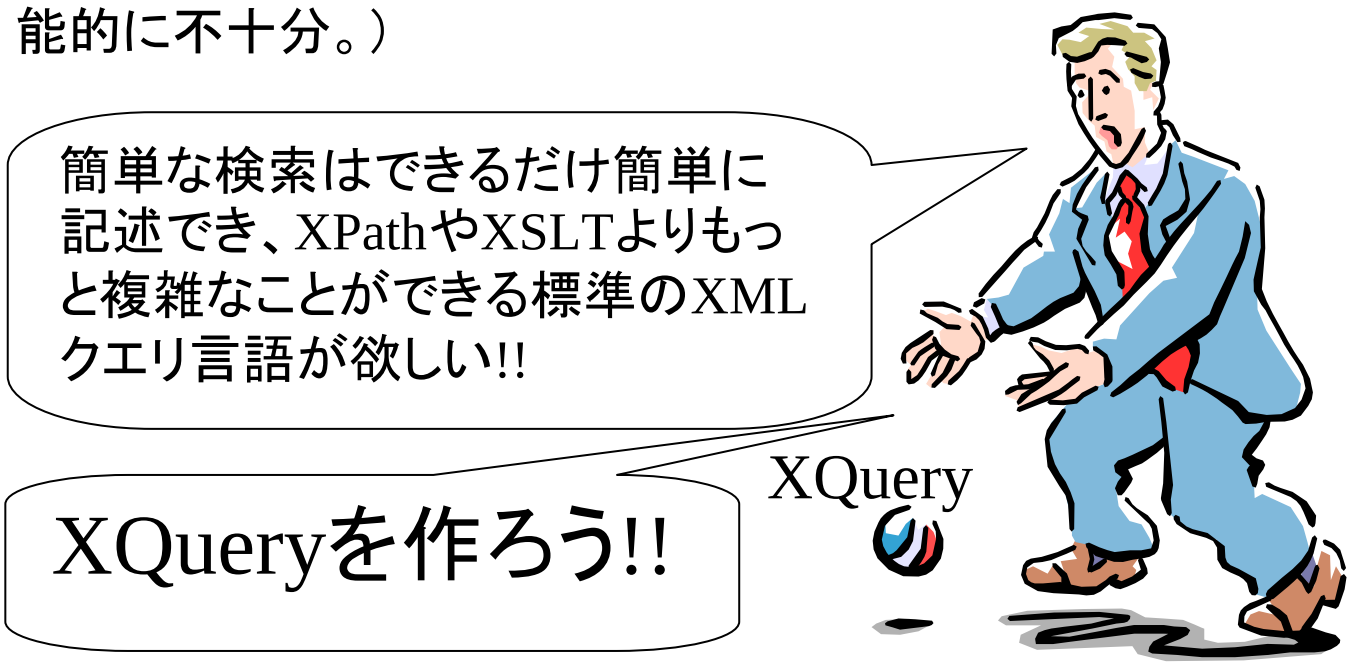
XQueryの登場の背景

- XMLネイティブデータベースのクエリ言語
  - XMLネイティブデータベースに対する標準のクエリ言語としてはXPathがある。（しかし、機能的にまだ不満が残る。）
  - XSLTを標準のクエリ言語として使うことは出来ないか？

確かにXSLTであれば、XPathではできなかったノードを組み合わせて新しいXMLのデータ構造を作ることができます。
しかし、XSLTでも機能的に十分とはいえません。
また、クエリ言語としては、もっと簡単に記述できるものが望ましいと思います。

XQueryの登場の背景

- XMLネイティブデータベースのクエリ言語
  - XMLネイティブデータベースに対する標準のクエリ言語としてはXPathがある。（しかし、機能的にまだ不満が残る。）
  - XSLTを標準のクエリ言語として使うことは出来ないか？（まだ機能的に不十分。）



XMLクエリ言語に求められるもの

- XMLで表現できる様々なデータモデルへの検索
  - XMLクエリ言語は、XMLで表現できる様々なデータモデルに対して適切な検索ができなければならない。

さまざまなデータモデル

- 階層モデル、ネットワークモデル
- リレーショナルモデル
- オブジェクトモデル
- 半構造モデル

XMLクエリ言語に求められるもの

- XMLで表現できる様々なデータモデルへの検索
  - XMLクエリ言語は、XMLで表現できる様々なデータモデルに対して適切な検索ができなければならない。

ここで、例としてリレーショナルモデルを取り上げてみたいと思います。
リレーショナルモデルを実装したリレーショナルデータベースでは、SQLを使うことによって様々な検索を行うことが出来ます。
では、リレーショナルモデルのデータを表現したXMLに対して検索を行う場合を考えてみましょう。
当然、SQLでできる検索機能は全部出来てほしいはずです。

XMLクエリ言語に求められるもの

- XMLで表現できる様々なデータモデルへの検索
  - XMLクエリ言語は、XMLで表現できる様々なデータモデルに対して適切な検索ができなければならない。

このように、XMLクエリ言語はSQLの検索機能を包含しているべきです。
また、他のデータモデルについても、同様のことがいえます。

XQueryはこれらのデータモデルに対して適切に検索ができるように考えられています。

XMLクエリ言語に求められるもの

- 最適な検索性能を出せるような表現能力
  - XMLクエリ言語の記述能力によって、高速な検索を妨げるようなことはあってはならない。

XMLクエリ言語は、最適な検索性能が出せるような表現能力を持つ必要があります。

例えば、結合処理(Join)を行いたいとき、『直積を行ってから選択を行え！！』という指示しか出せないとしたらどうでしょう。
もっと高速な方法で結合処理を行いたくても、実装できないことになります。

XMLクエリ言語に求められるもの

- 最適な検索性能を出せるような表現能力
  - XMLクエリ言語の記述能力によって、高速な検索を妨げるようなことはあってはならない。

また、XMLをリレーショナルデータベースに格納して、その部分XMLを検索したい場合などを考えます。
XMLはもともとノードの順序を保証していますが、検索結果として順序を気にしないで検索したい場合、順序無保証のオプションがあると高速に検索できるかもしれません。
普通、リレーショナルデータベースで順序を保証するにはソートを行いますが、順序無保証であれば、ソートを行う必要はありません。

XMLクエリ言語に求められるもの

- 最適な検索性能を出せるような表現能力
  - XMLクエリ言語の記述能力によって、高速な検索を妨げるようなことはあってはならない。

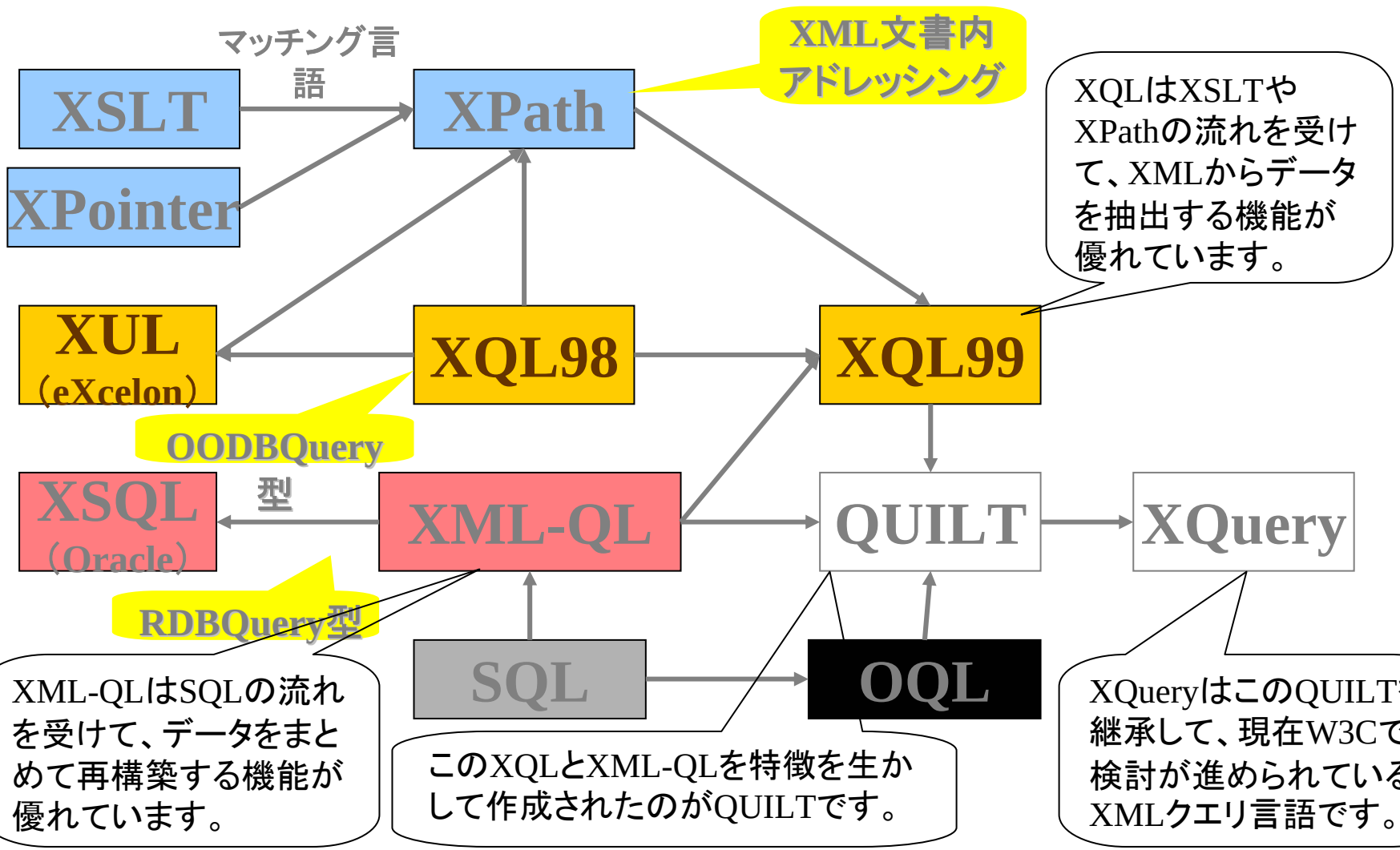
XQueryでは、順序無保証で検索したい場合には、unorderedというオプションを使えば良いことになっています。

(例) unorderedオプションの使用例

```
unordered(
 for $p in doc("parts.xml")//part[color = "Red"],
       $s in doc("suppliers.xml")//supplier
 where $p/suppno = $s/suppno
 return <ps> { $p/partno, $s/suppno } </ps> )
```

このように、XQueryは性能に関する問題も考慮して検討されています。

XML Query の流れ



XQuery W3C Working Draft

(www.w3.org/XML/Query)

- XML Query Requirements (5/2/2003)
- XML Query Use Cases (5/2/2003)
  （日本語訳:11/15/2002版）
  http://www.xmlconsortium.org/wg/tech/WD-xmlquery-use-cases-20021115-Japan-clean.htm
- XQuery1.0 and XPath2.0 Data Model (LAST CALL) (5/2/2003)
- XSLT 2.0 and XQuery 1.0 Serialization (5/2/2003)
- XQuery1.0 and XPath2.0 Formal Semantics (5/2/2003)
- XQuery1.0: An XML Query Language (5/2/2003)
- XML Syntax for XQuery1.0 (XQueryX) (6/7/2001)
- XQuery1.0 and XPath2.0 Functions and Operators (LAST CALL)(5/2/2003)
- XPath Requirements Version2.0 (2/14/2001)
- XML Path Language (XPath)2.0 (5/2/2003)
  （日本語訳:8/16/2002版）
  http://www.xmlconsortium.org/wg/tech/WD-xpath20-20020816-Japan-without-Appendix.htm
- XML Query and XPath Full-Text Requirements (5/2/2003)
- XML Query and XPath Full-Text Use Cases (2/14/2003)

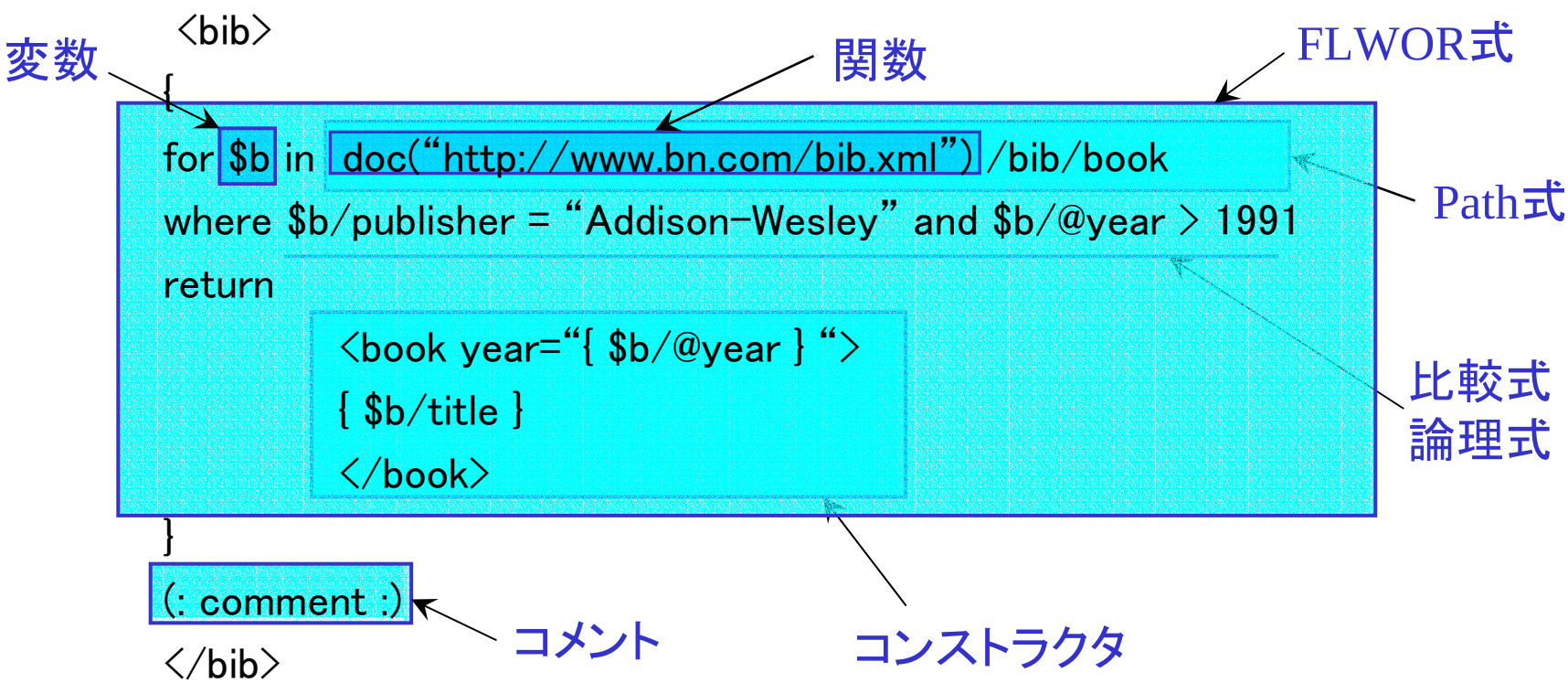
青字のDocumentは当WGが日本語訳し、公開しています。

緑字のDocumentは現在、当WGで日本語訳中です。

XQueryの基本

- クエリは式（Expression）の組み合わせで構成される

XQueryの式の例（2003/05/03 Working Draftより）



変数とPath式

- 変数
  - 「$」＋QName
    - 例：$b、$val
  - 式の結果であるシーケンス（評価コンテキスト）がバインドされる
    - FLWOR式、量化式、typeswitch式を使用
- Path式
  - 「XML Path Language (XPath)2.0」で定義
  - ツリー内のノードの位置を示すために使用される
  - 「ステップ」によりシーケンスが作成され、「述語」により取捨選択される
  - 例）/doc/chapter[title="Introduction"]

関数と演算子

- XQuery内で使用される関数と演算子
  - 「XQuery1.0 and XPath2.0 Functions and Operators」で定義
  - 関数の例:
    - 数値に関する関数と演算子
      - op:numeric-add、fn:round など
    - ストリング関数
      - fn:concat、fn:contains、fn:replace など
    - ノードやシーケンスを扱う関数と演算子
      - fn:local-name、fn:deep-equal、fn:empty、fn:insert-before、fn:sum、fn:docなど
    - その他、コンストラクタ関数、日付や時間に関する関数、コンテキスト関数、キャスト関数など多数

コンストラクタ

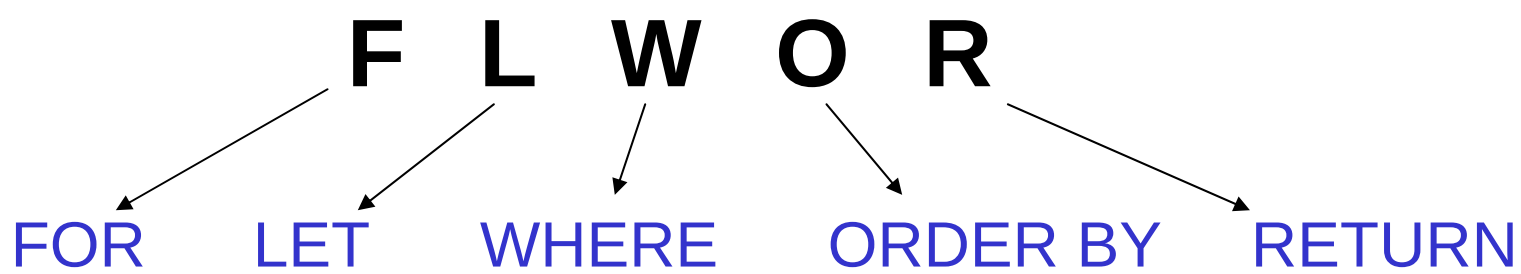
- XMLを新しく作り出す式
  - Direct Element Constructor
    - 文字列（リテラル）はそのまま生成されるXMLの一部となる
    - curly braces「{ }」内は式として評価され、結果と置き換えられる。
    - 例）

```
<title isbn="isbn-0060229357">
    { $b/title }
</title>
```

  - Computed Constructor
    - キーワード（element、attribute、document、text）の後にノード名を指定することで、そのノードを生成する
    - 例）

```
element title { "Harold and the Purple Crayon" }
    {
        attribute isbn { "isbn-0060229357" }
    }
```

FLWOR式（1）



- flower [fləʊə(r)] と読む
- 複数のXMLドキュメントのJOINやXMLデータの再構築が可能
- 入れ子（FLWOR式の中にFLWOR式を入れること）も可能

FLWOR式（2）

- FOR句

  for 変数 in シーケンス (, 変数 in シーケンス)*

  - シーケンスのCartesian積（直積）の組を変数にバインドし、変数の組（tuple stream）を作成する
  - シーケンス内の各要素に対して繰り返し処理（iteration）を行う
- LET句

  let 変数 := シーケンス (, 変数 in シーケンス)*

  - シーケンス全体を直接変数にバインドし、tuple streamを作成する
  - FOR句がある場合には、FOR句で作成された変数の組ごとに作成され、FOR句がない場合には、ひとつの変数が作成される

FLWOR式（3）

- WHERE句

  where 条件式

  - FOR句やLET句で作成された変数の組（tuple stream）をフィルターする
  - 条件がtrueの変数は残され、falseの変数は除かれる
- ORDER BY句

  order by 変数（シーケンス）

  - tuple stream の順序を並びかえる
  - オプションとして、並び替えの順序やcollation、emptyシーケンスの扱い等を指定する

FLWOR式（4）

- RETURN句

  return コンストラクタ

  - FLWOR式の結果を作成する
  - tuple stream内のそれぞれの変数に対して、1回ずつ評価（計算）される

XML Query Use Cases

- 9つの記述例を記載
  - 記述例(“XMP”):代表例
  - 記述例(“TREE”):階層構造を保持するクエリ
  - 記述例(“SEQ”):シーケンスにもとづくクエリ
  - 記述例(“R”):リレーショナルデータに対するアクセス
  - 記述例("SGML"):Standard Generalized Markup Language
  - 記述例(“STRING”):ストリング検索
  - 記述例(“NS”):名前空間を使用したクエリ
  - 記述例(“PARTS”):再帰的部分展開
  - 記述例("STRONG"):強く型指定されたデータを利用するクエリ

XML Query Use Cases の具体例

- Use Case 例1
  - 「1991年以降にAddison-Wesleyによって出版された本の、出版年とタイトルをリストする」 (Use Case XMP Q1)
- Use Case 例2
  - 「すべての自転車（Bicycle）について、アイテム番号（itemno）、説明（description）および（もしあれば）最も高い入札価格（bid）をリストにし、アイテム番号順に並べ替える」（Use Cases R Q2）

Use Case 例1

- 「1991年以降にAddison-Wesleyによって出版された本の、出版年とタイトルをリストする」(Use Case XMP Q1)

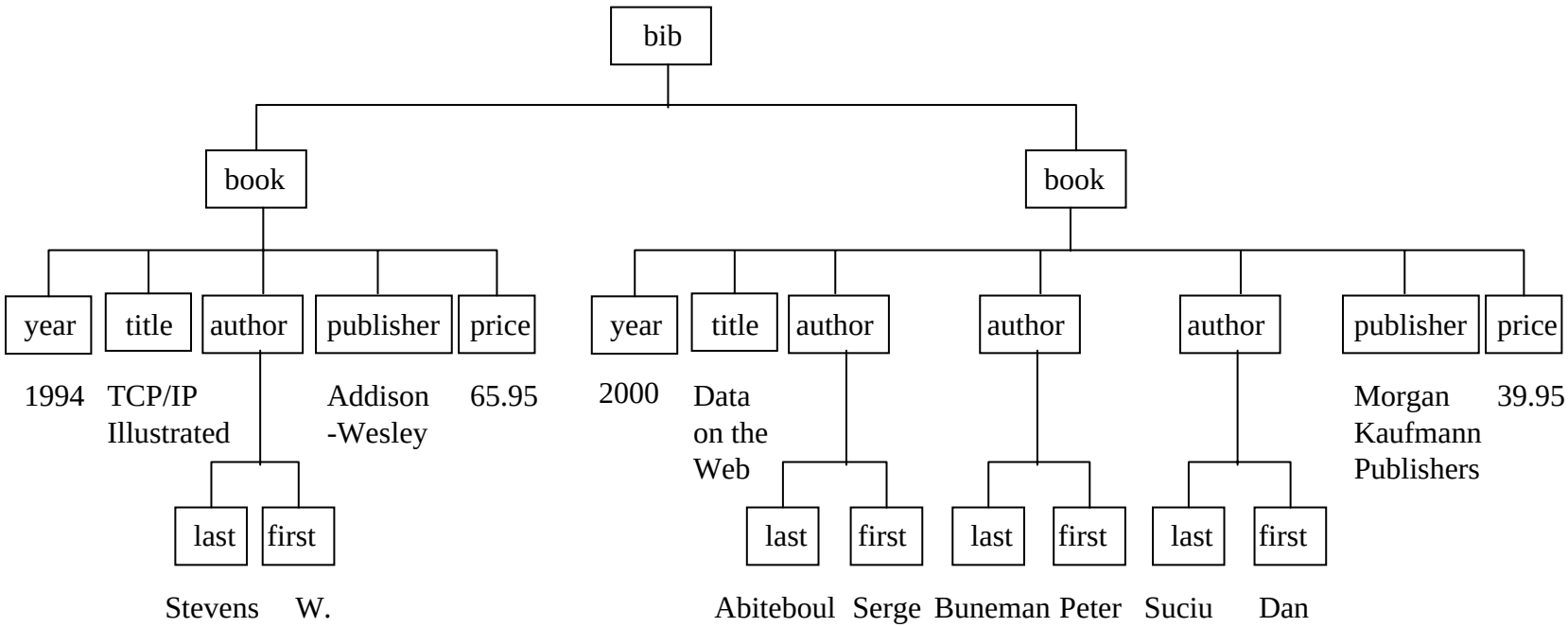
サンプルXMLデータ（bib.xml）

```
<bib>
 <book year="1994">
  <title>TCP/IP Illustrated</title>
  <author>
   <last>Stevens</last>
   <first>W.</first>
  </author>
  <publisher>Addison-Wesley</publisher>
  <price> 65.95</price>
 </book>
```

```
 <book year="2000">
  <title>Data on the Web</title>
  <author>
   <last>Abiteboul</last>
   <first>Serge</first>
  </author>
  <author>
   <last>Buneman</last>
   <first>Peter</first>
  </author>
  <author>
   <last>Suciu</last>
   <first>Dan</first>
  </author>
  <publisher>Morgan Kaufmann Publishers</publisher>
  <price>39.95</price>
 </book>
</bib>
```

XML Query Use Cases に記載されているデータを
参考に、一部データを省略しています

bib.xml のツリー構造



XQueryの例1

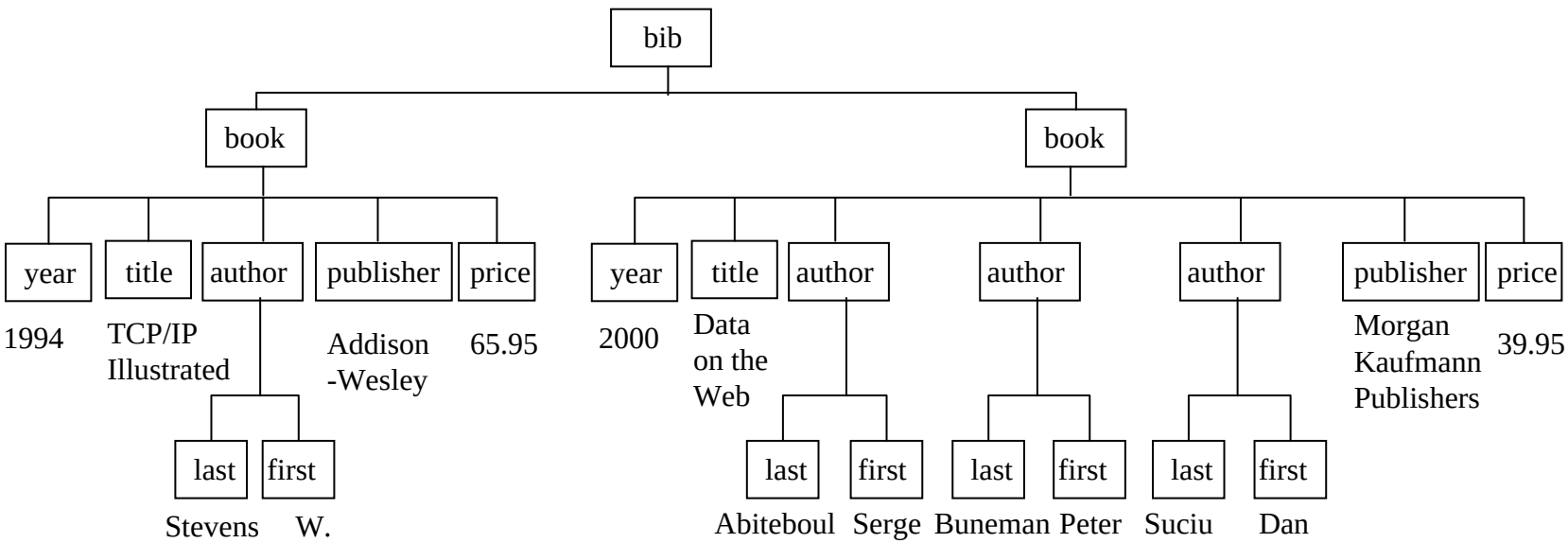
- 「1991年以降にAddison-Wesleyによって出版された本の、出版年とタイトルをリストする」(Use Case XMP Q1)

XQueryの記述例

```
 <bib>
{
for $b in  doc("http://www.bn.com/bib.xml") /bib/book
where $b/publisher = "Addison-Wesley" and $b/@year > 1991
return
            <book year="{ $b/@year } ">
            { $b/title }
            </book>
}
</bib>
```

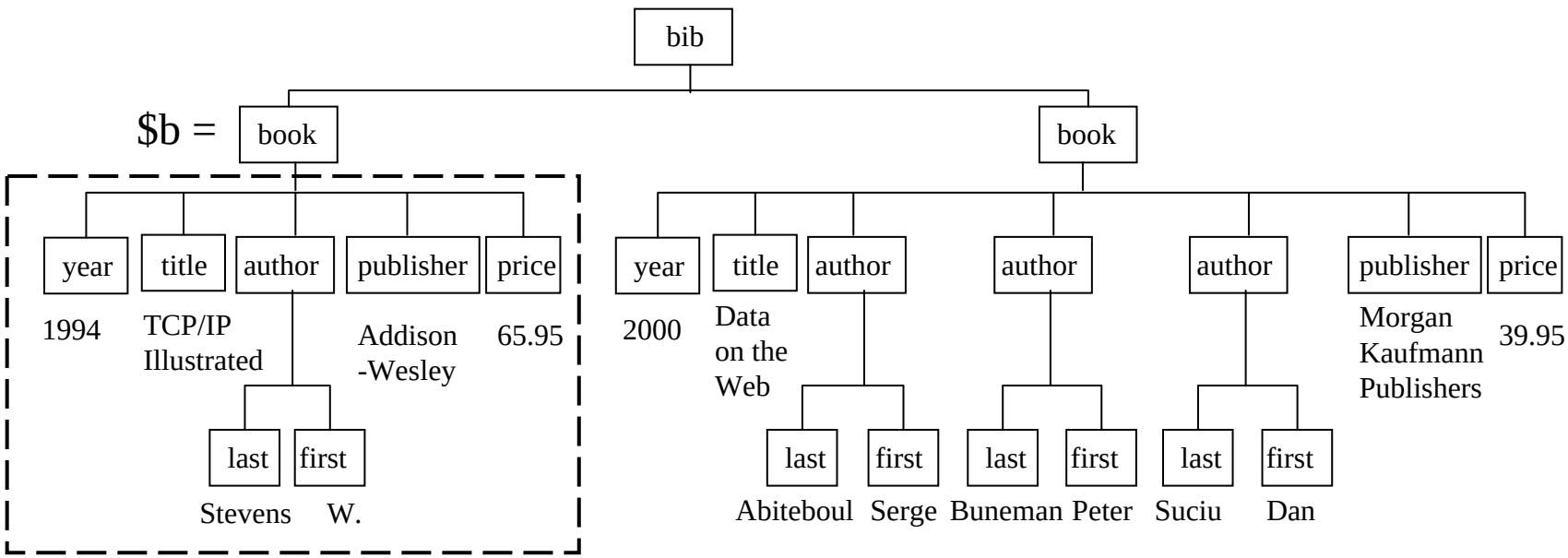
XQueryの処理の解説1（1）

```
<bib>
{
for $b in  doc("http://www.bn.com/bib.xml") /bib/book
where $b/publisher = "Addison-Wesley" and $b/@year > 1991
return
                <book year="{ $b/@year } ">
                { $b/title }
                </book>
}
</bib>
```



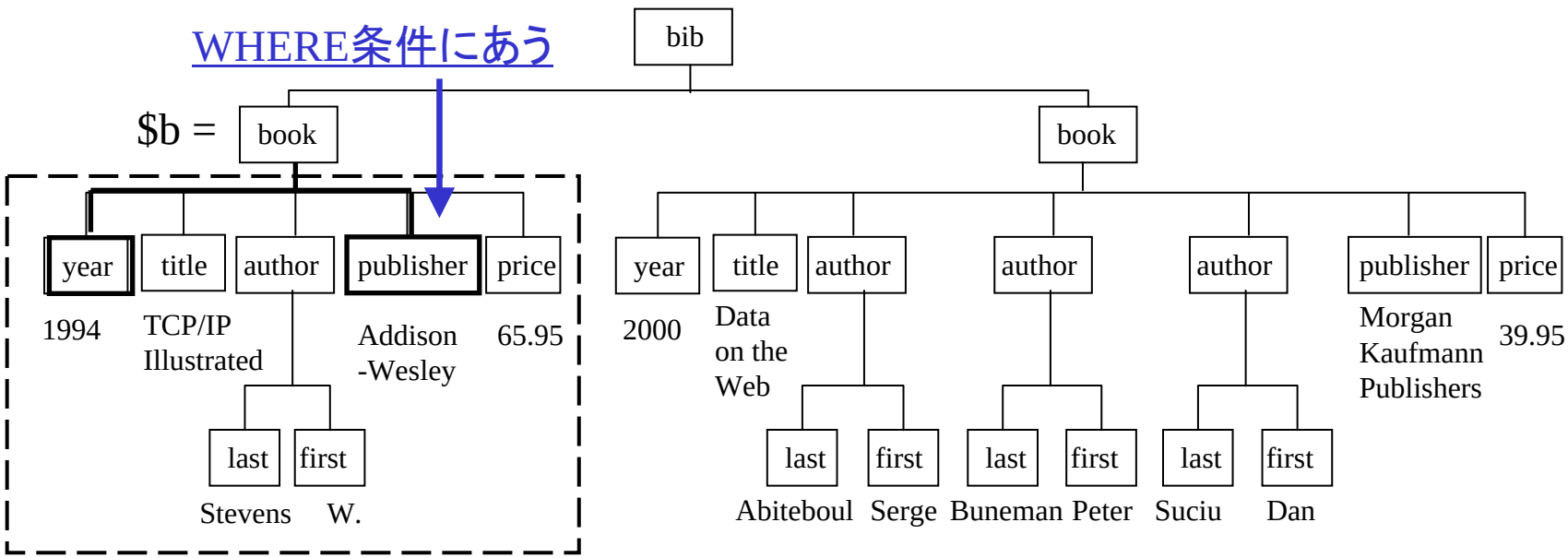
XQueryの処理の解説1（2）

```
<bib>
{
for $b in  doc("http://www.bn.com/bib.xml") /bib/book
where $b/publisher = "Addison-Wesley" and $b/@year > 1991
return
            <book year="{ $b/@year } ">
            { $b/title }
            </book>
}
</bib>
```



XQueryの処理の解説1（3）

```
<bib>
{
for $b in  doc("http://www.bn.com/bib.xml") /bib/book
where $b/publisher = "Addison-Wesley" and $b/@year > 1991
return
            <book year="{ $b/@year } ">
            { $b/title }
            </book>
}
</bib>
```

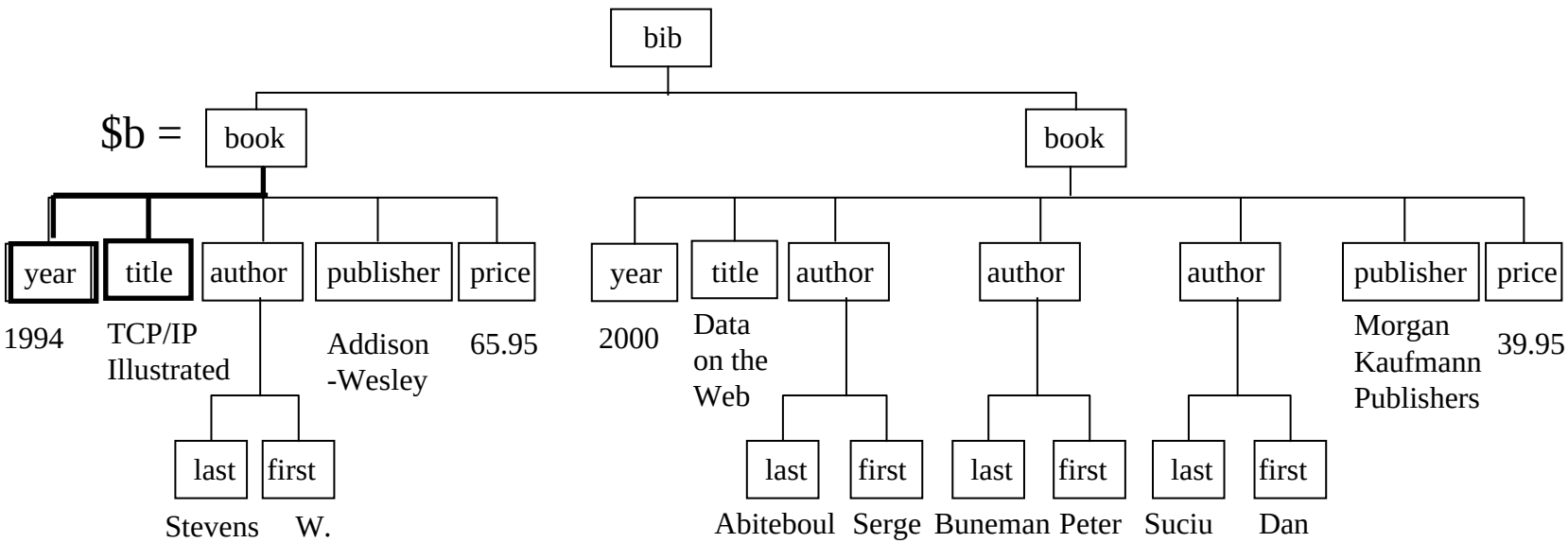


XQueryの処理の解説1（4）

```
<bib>
{
for $b in  doc("http://www.bn.com/bib.xml") /bib/book
where $b/publisher = "Addison-Wesley" and $b/@year > 1991
return
        <book year="{ $b/@year } ">
        { $b/title }
        </book>
}
</bib>
```

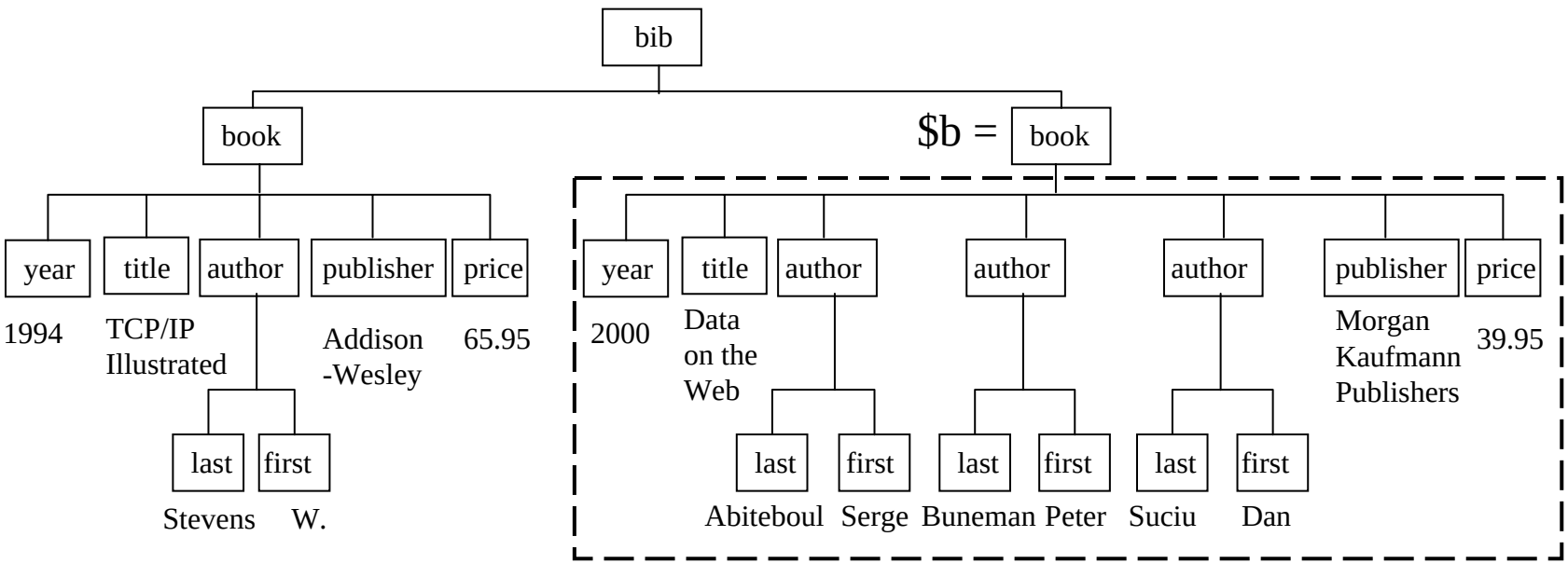
return句の結果

```
<book year="1994">
  <title>TCP/IP Illustrated</title>
</book>
```



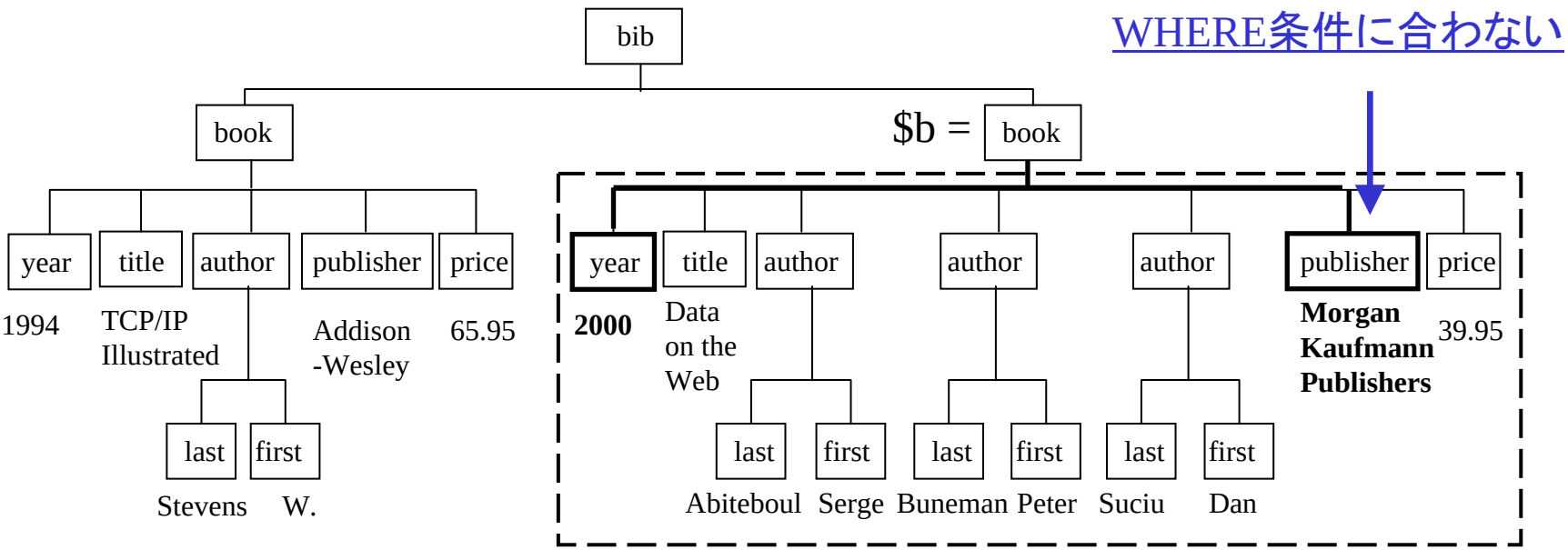
XQueryの処理の解説1（5）

```
<bib>
{
for $b in  doc("http://www.bn.com/bib.xml") /bib/book
where $b/publisher = "Addison-Wesley" and $b/@year > 1991
return
            <book year="{ $b/@year } ">
            { $b/title }
            </book>
}
</bib>
```



XQueryの処理の解説1(6)

```
<bib>
{
for $b in  doc("http://www.bn.com/bib.xml") /bib/book
where $b/publisher = "Addison-Wesley" and $b/@year > 1991
return
                <book year="{ $b/@year } ">
                { $b/title }
                </book>
}
</bib>
```

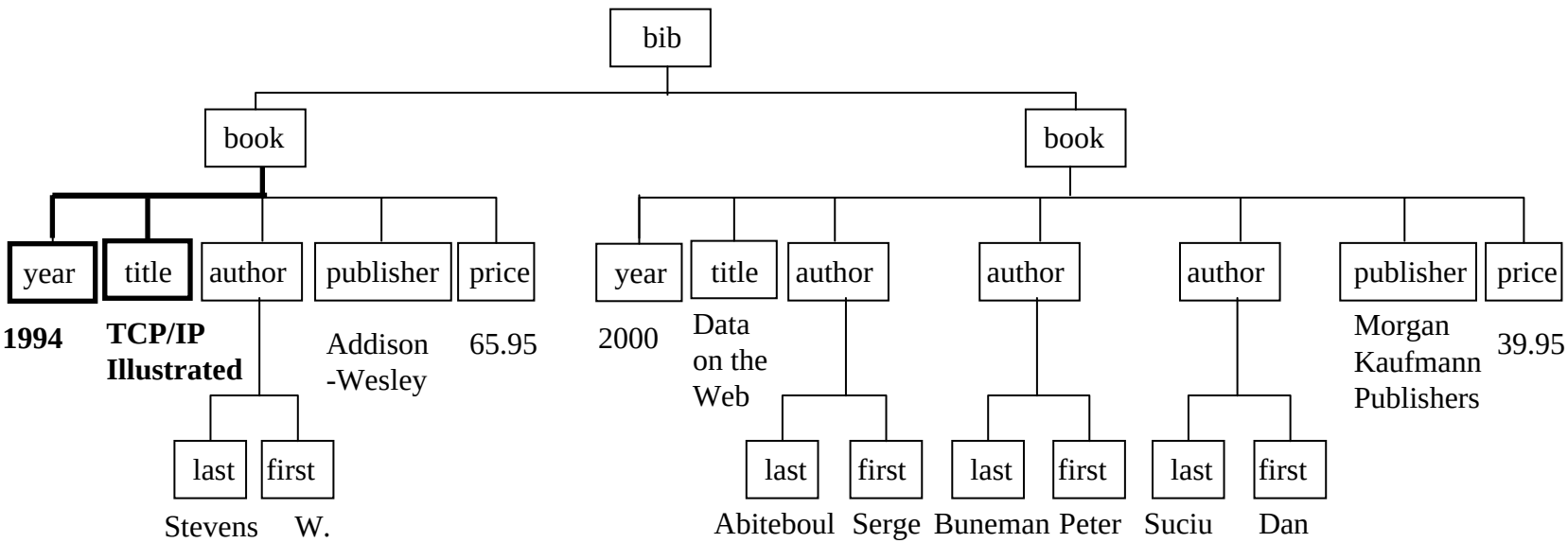


XQueryの処理の解説1（7）

```
 <bib>
{
for $b in  doc("http://www.bn.com/bib.xml") /bib/book
where $b/publisher = "Addison-Wesley" and $b/@year > 1991
return
                <book year="{ $b/@year } ">
                { $b/title }
                </book>
}
</bib>
```

XQueryの最終的な結果

```
<bib>
  <book year="1994">
    <title>TCP/IP Illustrated</title>
  </book>
</bib>
```



Use Case 例2

- 「すべての自転車（Bicycle）について、アイテム番号（itemno）、説明（description）および（もしあれば）最も高い入札価格（bid）をリストにし、アイテム番号順に並べ替える」（Use Cases R Q2）

サンプル・リレーショナル・データ

ITEMS表

ITEMNO	DESCRIPTION	OFFERED_BY	START_DATE	END_DATE	RESERVE_PRICE
1001	Red Bicycle	U01	1999-01-05	1999-01-20	40
1002	Motorcycle	U02	1999-02-11	1999-03-15	500
1003	Old Bicycle	U02	1999-01-10	1999-02-20	25
1004	Tricycle	U01	1999-02-25	1999-03-08	15
1005	Tennis Racket	U03	1999-03-19	1999-04-30	20
1006	Helicopter	U03	1999-05-05	1999-05-25	50000
1007	Racing Bicycle	U04	1999-01-20	1999-02-20	200
1008	Broken Bicycle	U01	1999-02-05	1999-03-06	25

BIDS表

USERID	ITEMNO	BID	BID_DATE
U02	1001	35	1999-01-07
U04	1001	40	1999-01-08
U02	1001	45	1999-01-11
U04	1001	50	1999-01-13
U02	1001	55	1999-01-15
U01	1002	400	1999-02-14
U02	1002	600	1999-02-16
U03	1002	800	1999-02-17
U04	1002	1000	1999-02-25
U02	1002	1200	1999-03-02
U04	1003	15	1999-01-22
U05	1003	20	1999-02-03
U01	1004	40	1999-03-05
U03	1007	175	1999-01-25
U05	1007	200	1999-02-08
U04	1007	225	1999-02-12

サンプルXMLデータ（items.xml）

items.xml

```
<items>
<item_tuple><itemno>1001</itemno><description>Red Bicycle</description><offered_by>U01</offered_by>
        <start_date>1999-01-05</start_date><end_date>1999-01-20</end_date><reserve_price>40</reserve_price></item_tuple>
<item_tuple><itemno>1002</itemno><description>Motorcycle</description><offered_by>U02</offered_by>
        <start_date>1999-02-11</start_date><end_date>1999-03-15</end_date><reserve_price>500</reserve_price></item_tuple>
<item_tuple><itemno>1003</itemno><description>Old Bicycle</description><offered_by>U02</offered_by>
        <start_date>1999-01-10</start_date><end_date>1999-02-20</end_date><reserve_price>25</reserve_price></item_tuple>
<item_tuple><itemno>1004</itemno><description>Tricycle</description><offered_by>U01</offered_by>
        <start_date>1999-02-25</start_date><end_date>1999-03-08</end_date><reserve_price>15</reserve_price></item_tuple>
<item_tuple><itemno>1005</itemno><description>Tennis Racket</description><offered_by>U03</offered_by>
        <start_date>1999-03-19</start_date><end_date>1999-04-30</end_date><reserve_price>20</reserve_price></item_tuple>
<item_tuple><itemno>1006</itemno><description>Helicopter</description><offered_by>U03</offered_by>
        <start_date>1999-05-05</start_date><end_date>1999-05-25</end_date><reserve_price>50000</reserve_price></item_tuple>
<item_tuple><itemno>1007</itemno><description>Racing Bicycle</description><offered_by>U04</offered_by>
        <start_date>1999-01-20</start_date><end_date>1999-02-20</end_date><reserve_price>200</reserve_price></item_tuple>
<item_tuple><itemno>1008</itemno><description>Broken Bicycle</description><offered_by>U01</offered_by>
        <start_date>1999-02-05</start_date><end_date>1999-03-06</end_date><reserve_price>25</reserve_price></item_tuple>
</items>
```

サンプルXMLデータ（bids.xml）

bids.xml

```
<bids>
<bid_tuple> <userid>U02</userid> <itemno>1001</itemno> <bid>35</bid>   <bid_date>1999-01-07</bid_date> </bid_tuple>
<bid_tuple> <userid>U04</userid> <itemno>1001</itemno> <bid>40</bid>   <bid_date>1999-01-08</bid_date > </bid_tuple>
<bid_tuple> <userid>U02</userid> <itemno>1001</itemno> <bid>45</bid>   <bid_date>1999-01-11</bid_date> </bid_tuple>
<bid_tuple> <userid>U04</userid> <itemno>1001</itemno> <bid>50</bid>   <bid_date>1999-01-13</bid_date > </bid_tuple>
<bid_tuple> <userid>U02</userid> <itemno>1001</itemno> <bid>55</bid>   <bid_date>1999-01-15</bid_date> </bid_tuple>
<bid_tuple> <userid>U01</userid> <itemno>1002</itemno> <bid>400</bid>  <bid_date>1999-02-14</bid_date> </bid_tuple>
<bid_tuple> <userid>U02</userid> <itemno>1002</itemno> <bid>600</bid>  <bid_date>1999-02-16</bid_date> </bid_tuple>
<bid_tuple> <userid>U03</userid> <itemno>1002</itemno> <bid>800</bid>  <bid_date>1999-02-17</bid_date> </bid_tuple>
<bid_tuple> <userid>U04</userid> <itemno>1002</itemno> <bid>1000</bid> <bid_date>1999-02-25</bid_date> </bid_tuple>
<bid_tuple> <userid>U02</userid> <itemno>1002</itemno> <bid>1200</bid> <bid_date>1999-03-02</bid_date> </bid_tuple>
<bid_tuple> <userid>U04</userid> <itemno>1003</itemno> <bid>15</bid>   <bid_date>1999-01-22</bid_date> </bid_tuple>
<bid_tuple> <userid>U05</userid> <itemno>1003</itemno> <bid>20</bid>   <bid_date>1999-02-03</bid_date > </bid_tuple>
<bid_tuple> <userid>U01</userid> <itemno>1004</itemno> <bid>40</bid>   <bid_date>1999-03-05</bid_date> </bid_tuple>
<bid_tuple> <userid>U03</userid> <itemno>1007</itemno> <bid>175</bid>  <bid_date>1999-01-25</bid_date> </bid_tuple>
<bid_tuple> <userid>U05</userid> <itemno>1007</itemno> <bid>200</bid>  <bid_date>1999-02-08</bid_date> </bid_tuple>
<bid_tuple> <userid>U04</userid> <itemno>1007</itemno> <bid>225</bid>  <bid_date>1999-02-12</bid_date> </bid_tuple>
</bids>
```

XQueryの例2

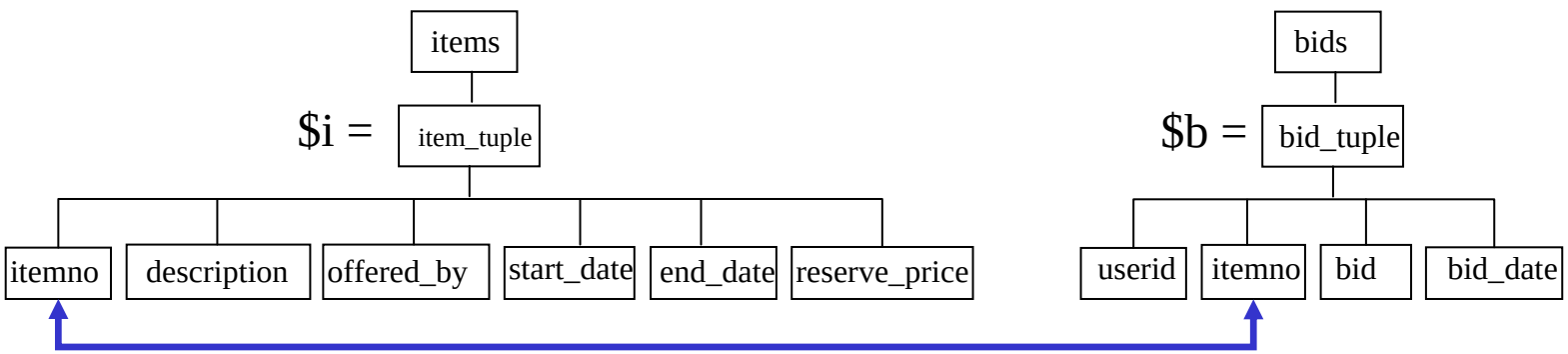
- 「すべての自転車（Bicycle）について、アイテム番号（itemno）、説明（description）および（もしあれば）最も高い入札価格（bid）をリストにし、アイテム番号順に並べ替える」（Use Cases R Q2）

XQueryの記述例

```
<result> {
    for $i in doc("items.xml")//item_tuple
    let $b := doc("bids.xml")//bid_tuple[itemno = $i/itemno]
    where contains($i/description, "Bicycle")
    order by $i/itemno
    return
        <item_tuple>
            { $i/itemno }
            { $i/description }
            <high_bid>{ max($b/bid) }</high_bid>
        </item_tuple>
  }</result>
```

XQueryの処理の解説2（1）

```
<result>  {
   for $i in doc("items.xml")//item_tuple
   let $b := doc("bids.xml")//bid_tuple[itemno = $i/itemno]
   where contains($i/description, "Bicycle")
   order by $i/itemno
   return
      <item_tuple>
         { $i/itemno }
         { $i/description }
         <high_bid>{ max($b/bid) }</high_bid>
      </item_tuple>
 }</result>
```



XMLデータのJOIN

XQueryの処理の解説2（2）

```
<result>  {
   for $i in doc("items.xml")//item_tuple
   let $b := doc("bids.xml")//bid_tuple[itemno = $i/itemno]
   where contains($i/description, "Bicycle")
   order by $i/itemno
   return
      <item_tuple>
         { $i/itemno }
         { $i/description }
         <high_bid>{ max($b/bid) }</high_bid>
      </item_tuple>
 }</result>
```

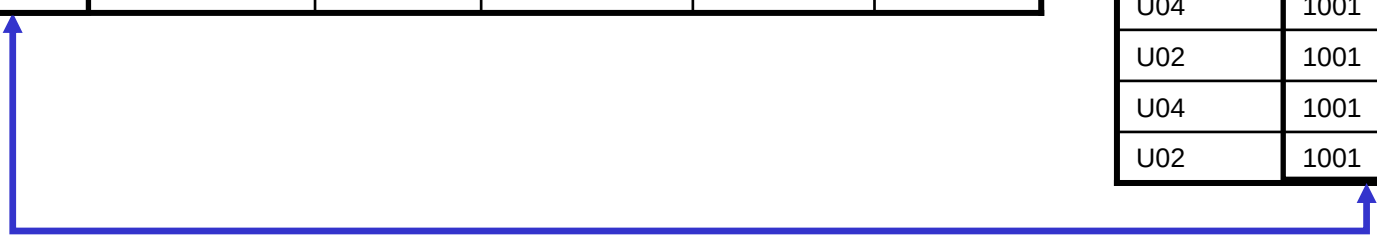
表データでJOINを考えると・・・

$i =

ITEMNO	DESCRIPTION	OFFERED_BY	START_DATE	END_DATE	RESERVE_PRICE
1001	Red Bicycle	U01	1999-01-05	1999-01-20	40

$b =

USERID	ITEMNO	BID	BID_DATE
U02	1001	35	1999-01-07
U04	1001	40	1999-01-08
U02	1001	45	1999-01-11
U04	1001	50	1999-01-13
U02	1001	55	1999-01-15



XQueryの処理の解説2（3）

```
<result>  {
   for $i in doc("items.xml")//item_tuple
   let $b := doc("bids.xml")//bid_tuple[itemno = $i/itemno]
   where contains($i/description, "Bicycle")
   order by $i/itemno
   return
      <item_tuple>
         { $i/itemno }
         { $i/description }
         <high_bid>{ max($b/bid) }</high_bid>
      </item_tuple>
 }</result>
```

$i =

ITEMNO	DESCRIPTION	OFFERED_BY	START_DATE	END_DATE	RESERVE_PRICE
1001	Red Bicycle	U01	1999-01-05	1999-01-20	40

$b =

USERID	ITEMNO	BID	BID_DATE
U02	1001	35	1999-01-07
U04	1001	40	1999-01-08
U02	1001	45	1999-01-11
U04	1001	50	1999-01-13
U02	1001	55	1999-01-15

XQueryの処理の解説2（4）

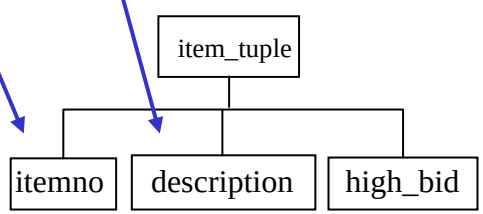
```
<result>  {
   for $i in doc("items.xml")//item_tuple
   let $b := doc("bids.xml")//bid_tuple[itemno = $i/itemno]
   where contains($i/description, "Bicycle")
   order by $i/itemno
   return
      <item_tuple>
         { $i/itemno }
         { $i/description }
         <high_bid>{ max($b/bid) }</high_bid>
      </item_tuple>
 }</result>
```

$i =

ITEMNO	DESCRIPTION	OFFERED_BY	START_DATE	END_DATE	RESERVE_PRICE
1001	Red Bicycle	U01	1999-01-05	1999-01-20	40

$b =

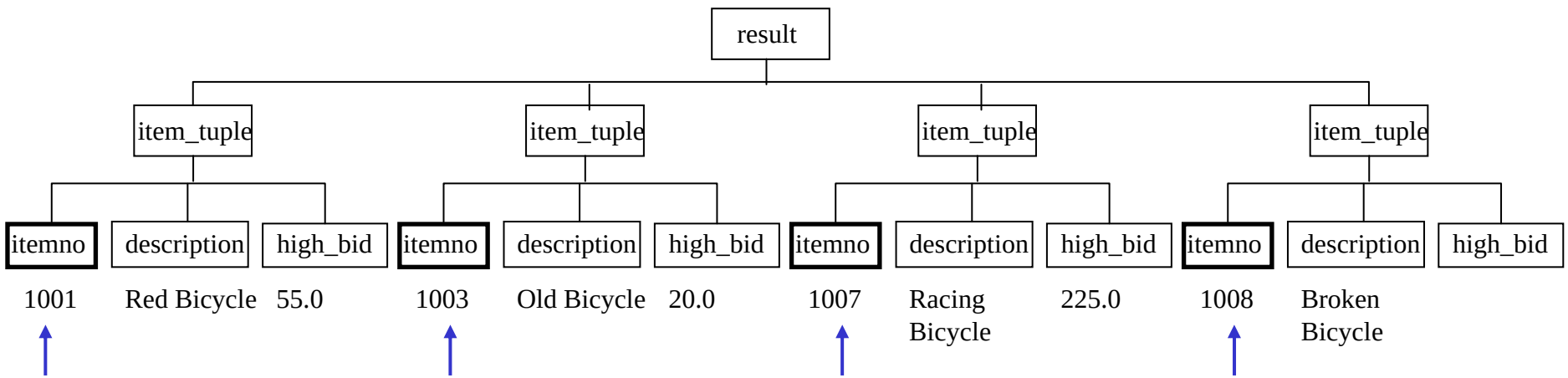
USERID	ITEMNO	BID	BID_DATE
U02	1001	35	1999-01-07
U04	1001	40	1999-01-08
U02	1001	45	1999-01-11
U04	1001	50	1999-01-13
U02	1001	55	1999-01-15



max( ) で最大値を取得

XQueryの処理の解説2（5）

```
<result>  {
   for $i in doc("items.xml")//item_tuple
   let $b := doc("bids.xml")//bid_tuple[itemno = $i/itemno]
   where contains($i/description, "Bicycle")
   order by $i/itemno
   return
      <item_tuple>
         { $i/itemno }
         { $i/description }
         <high_bid>{ max($b/bid) }</high_bid>
      </item_tuple>
 }</result>
```



order by により、昇順に並び替え

XQueryの処理の解説2（6）

- 結果

```
<result>
  <item_tuple>
    <itemno>1001</itemno>
    <description>Red Bicycle</description>
    <high_bid>55.0</high_bid>
  </item_tuple>
  <item_tuple>
    <itemno>1003</itemno>
    <description>Old Bicycle</description>
    <high_bid>20.0</high_bid>
  </item_tuple>
  <item_tuple>
    <itemno>1007</itemno>
    <description>Racing Bicycle</description>
    <high_bid>225.0</high_bid>
  </item_tuple>
  <item_tuple>
    <itemno>1008</itemno>
    <description>Broken Bicycle</description>
    <high_bid></high_bid>
  </item_tuple>
</result>
```

SVGデータをXQueryで検索する

- SVGとは？
  - Scalable Vector Graphics の略
  - W3Cで規格化されているベクターグラフィックスを記述するためのXMLベースの言語
    - http://www.w3.org/TR/SVG/
  - 簡単な図形の例）
    - rect 要素・・・四角形
    - circle 要素・・・円
    - text要素・・・文字列
- SVGに対するXQueryによる検索
  ＝検索の視覚化の実現

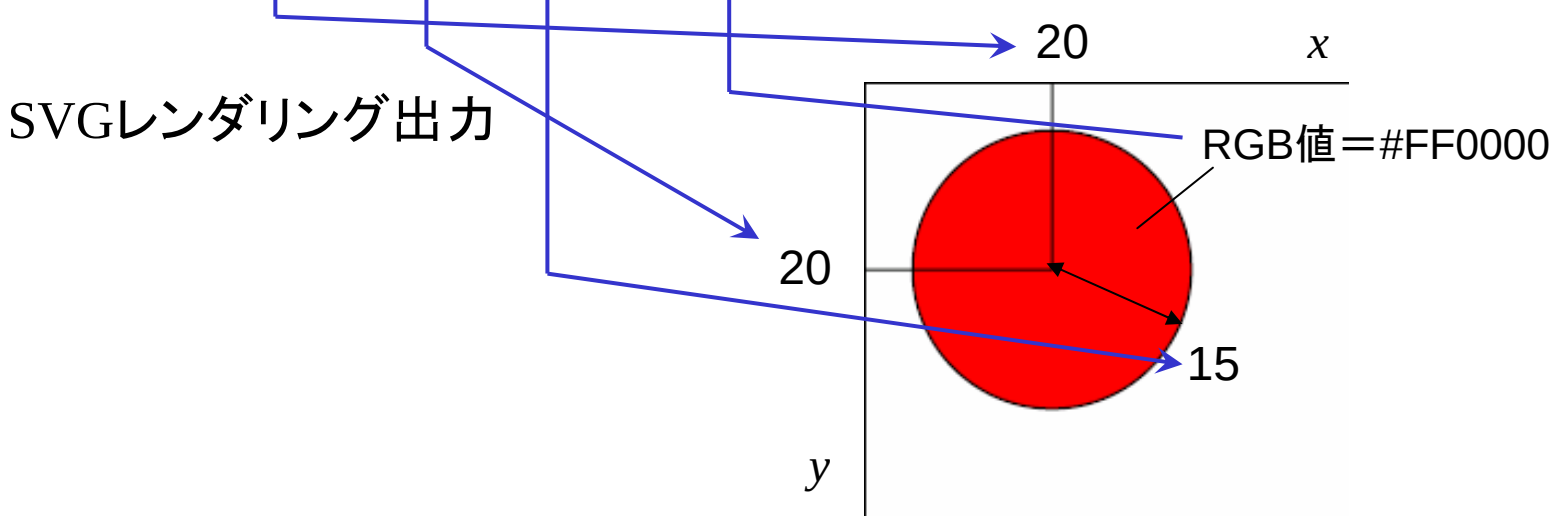
SVGの例

- 円の例

SVGソース

```
<?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>
```

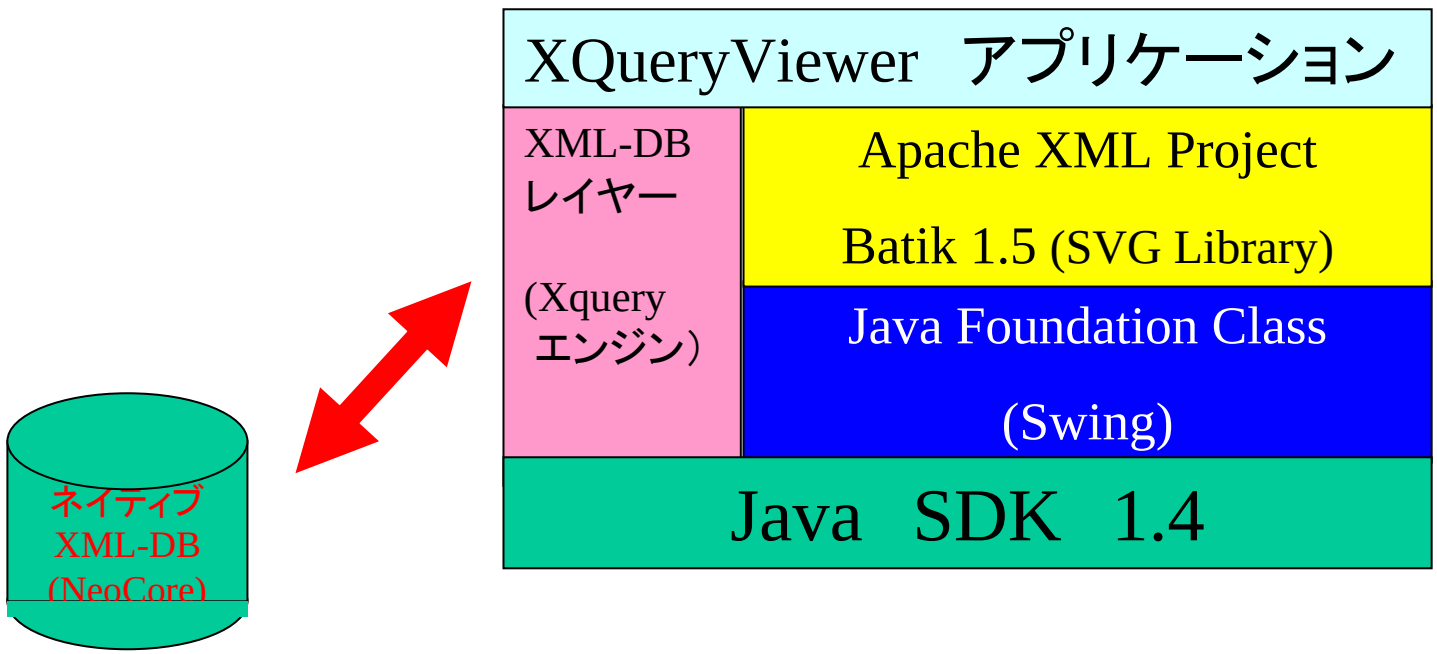
SVGレンダリング出力



デモ・プログラム概要

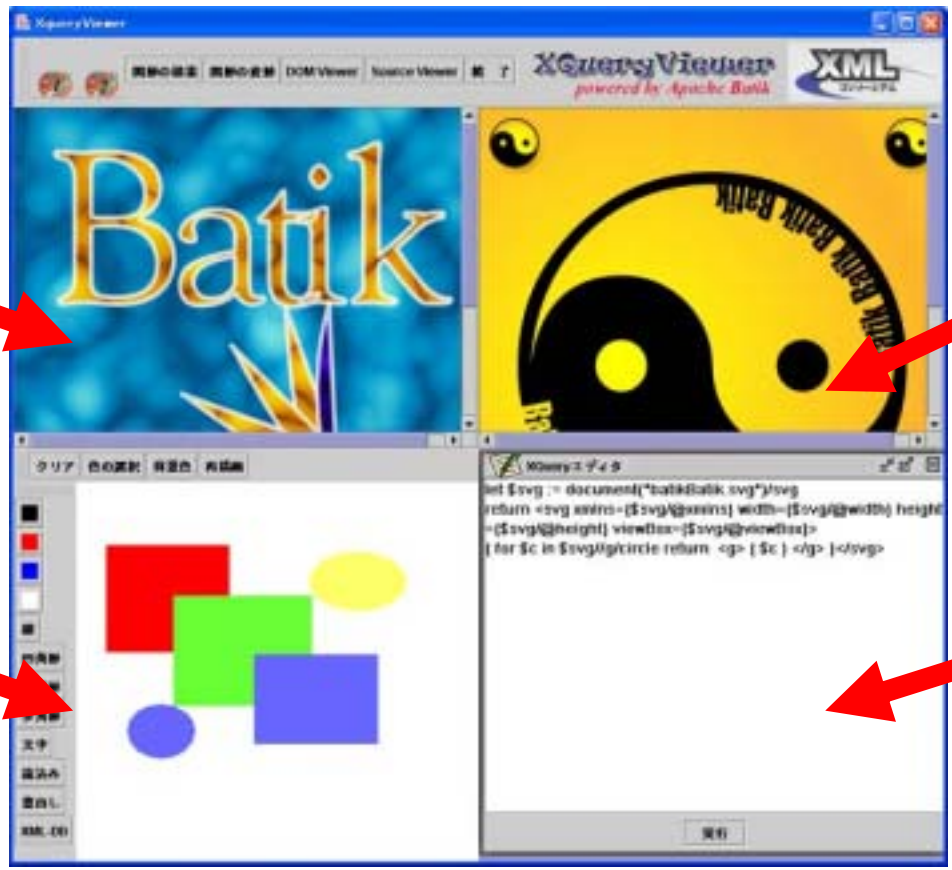
SVG

XQueryViewer
powered by Apache Batik



デモ・プログラム画面解説

XQueryViewer
powered by Apache Batik



SVGキャンバス－1

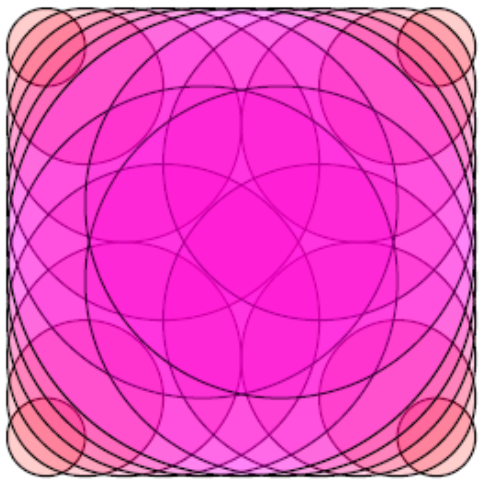
SVGキャンバス－2

落書きキャンバス

Xquery ビューワ

[参考] SVGに対するクエリ例（1）

- クエリ対象SVG（circle1.svg）
  - 半径が5、10、15、20、25、30の円が重なり合っている図形



[参考] SVGに対するクエリ例（2）

- SVGソース（circle1.svg）

```
<?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>
```

[参考] SVGに対するクエリ例（3）

- 半径が20より小さい円のみを表示するクエリ例

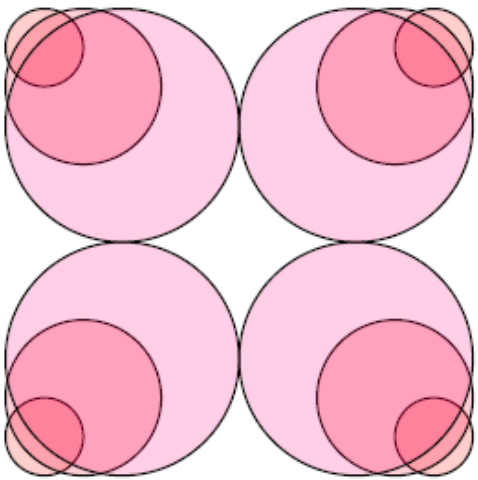
```
<svg xmlns='http://www.w3.org/2000/svg' width='100mm' height='100mm' viewBox='0 0 100 100'>
{
  for $c in document("circle1.svg")/svg//circle
  where $c/@r < 20
  return
    <g> { $c } </g>
}</svg>
```

- 半径が20より大きい円のみを表示するクエリ例

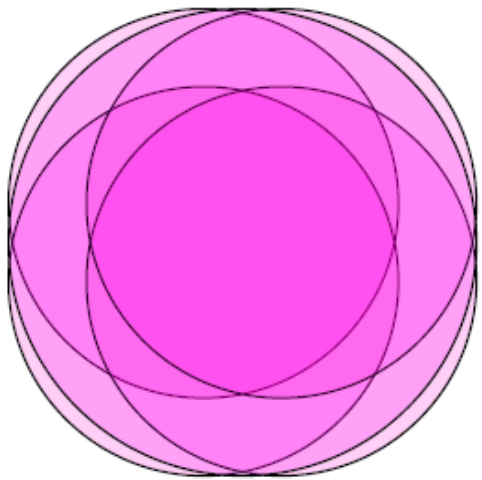
```
<svg xmlns='http://www.w3.org/2000/svg' width='100mm' height='100mm' viewBox='0 0 100 100'>
{
  for $c in document("circle1.svg")/svg//circle
  where $c/@r > 20
  return
    <g> { $c } </g>
}</svg>
```

[参考] SVGに対するクエリ例（4）

- 半径が20より小さいの円の検索結果

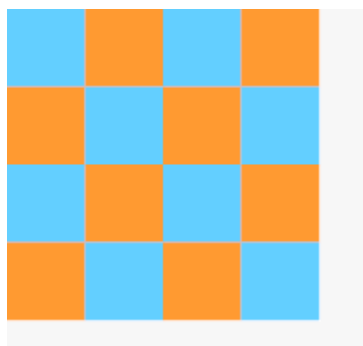


- 半径が20より大きいの円の検索結果



[参考] SVGに対するクエリ例（5）

- クエリ対象SVG（rect1.svg）
  - 縦10、横10のサイズの正方形が、4つ×4つに並んで、市松模様に色分けされている図形



[参考] SVGに対するクエリ例（6）

- SVGソース（circle1.svg）

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns='http://www.w3.org/2000/svg' width='40mm' height='40mm' viewBox='0 0 40 40'>
<g>
        <rect x='0' y='0' width='10' height='10' fill="#66CCFF" stroke="none"/>
        <rect x='10' y='0' width='10' height='10' fill="#FF9933" stroke="none"/>
        <rect x='20' y='0' width='10' height='10' fill="#66CCFF" stroke="none"/>
        <rect x='30' y='0' width='10' height='10' fill="#FF9933" stroke="none"/>
        <rect x='0' y='10' width='10' height='10' fill="#FF9933" stroke="none"/>
        <rect x='10' y='10' width='10' height='10' fill="#66CCFF" stroke="none"/>
        <rect x='20' y='10' width='10' height='10' fill="#FF9933" stroke="none"/>
        <rect x='30' y='10' width='10' height='10' fill="#66CCFF" stroke="none"/>
        <rect x='0' y='20' width='10' height='10' fill="#66CCFF" stroke="none"/>
        <rect x='10' y='20' width='10' height='10' fill="#FF9933" stroke="none"/>
        <rect x='20' y='20' width='10' height='10' fill="#66CCFF" stroke="none"/>
        <rect x='30' y='20' width='10' height='10' fill="#FF9933" stroke="none"/>
        <rect x='0' y='30' width='10' height='10' fill="#FF9933" stroke="none"/>
        <rect x='10' y='30' width='10' height='10' fill="#66CCFF" stroke="none"/>
        <rect x='20' y='30' width='10' height='10' fill="#FF9933" stroke="none"/>
        <rect x='30' y='30' width='10' height='10' fill="#66CCFF" stroke="none"/>
</g>
</svg>
```

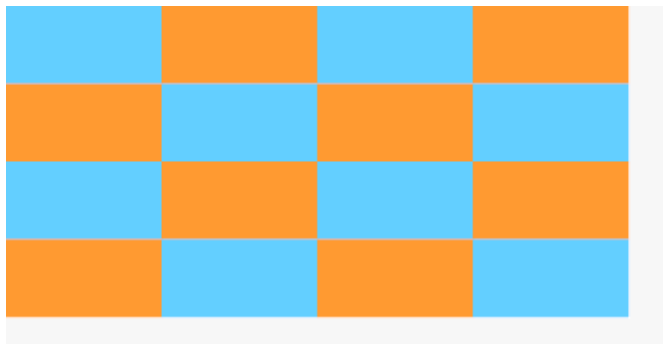
[参考] SVGに対するクエリ例（7）

- 対象図形を横方向（x方向）に2倍に拡大する図形を戻すクエリ

```
<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>
```

[参考] SVGに対するクエリ例（8）

- 検索結果



謝辞

今回のDemonstrationでは、三井物産様のご協力により、XMLネイティブDBとしてNeoCore（http://www.neocore.jp/）を使用しました。
また、SVGの地図データは、昭文社様にご提供いただきました。

ありがとうございました。