XML データベースへのコールバック関数の実装と適用

長瀬雅之[†1] 川口浩司[†2] 土井憲雄[†2] 中本啓之[†1] 飯田学[†1]

情報処理技術の進展に伴い，我々は膨大なデータを手にすることが可能になった．膨大に蓄積されたデータの解析・マイニングには，データベースシステムが不可欠であり，現在は主にリレーショナル・データベースが使用されている．一方で，今後は XML データの増大が予想され，XML データの高速解析のためのネイティブ XML データベースの必要性が増大すると思われるが，XML データベースには，現状いくつかの問題点がある．我々は，現状の問題点を解決するインメモリ XML データベース“Karearea”を開発し，これを宇宙航空研究開発機構 宇宙科学研究本部の SODA(SOlar Database & Archives)計画のプロトタイプシステムに適用した．我々が開発したインメモリ XML データベースは，独自の技術により，高速な検索や多次元集計が可能である．さらに，XML データに対しユーザが定義した複雑な学術計算などを行い，その結果をもとに検索することが可能な機能も実装した．これらの特徴を持つインメモリ XML データベースの本プロトタイプシステムへの適用とその評価を通して，我々は，データ解析システムのエンジンとしての XML データベース活用の可能性を見出した．

Implementation and application of the callback function mechanism to XML Database

MASAYUKI NAGASE,[†1] KOUJI KAWAGUCHI,[†2] NORIO DOI,[†2] HIROYUKI NAKAMOTO[†1] and MANABU IIDA[†1]

As information technology advances, it becomes possible for us to handle the large amount of data. Database management system is indispensable to perform a large amount of data analysis and data mining, and relational database management system (RDBMS) is mainly used for this purpose now. On the other hand, it is forecasted that the amount of XML data will increase and it becomes necessary to make the most use of a Native XML Database, but there are some problems in the current XML Database. Then, we have developed new In-Memory XML Database, "Karearea", to solve these problems. We applied Karearea to the prototype system of SODA (SOlar Database & Archives) by the Institute of Space and Astronautical Science. With our original technology, Karearea enabled not only high-speed searching, but also the multidimensional summary for XML data. And furthermore, we implemented a function to Karearea to search by the result of complicated, scientific and user-defined calculation to XML data. We found the possibility to make use of the XML database as an engine of the data analytic system through the application of this In-Memory Database to the prototype system and assessment of it.

1. はじめに

情報処理技術の進展に伴い，今日，我々は膨大なデータを手にすることが可能となった．そして，「データは蓄積する」から「活用する」という方向に技術，理論が進んでおり，データの解析，マイニング等の研究が盛んに行なわれている．しかし，膨大なデータの解析，マイニング等の情報処理には高速なデータ処理ツールが不可欠となるが，現状では，情報蓄積のスピードに比して，解析処理系の高速化技術の進展スピードは十分でないように思われる．解析処理系の高速化技術は，近年ではグリッドコンピューティングにその活路を見出そうとしているが，これまでは，スーパーコンピュータや，高価な専用ハードウェアに頼ることが必要であった．膨大なデータの中から新しい知見を発見しようとしている人たちは，各種研究機関の研究者，学生，新サービスを検討している企業人等，その裾野は広い．したがって，データ解析環境は，ごく普通のコンピュータ環境であることが望まれる．また，今後は，情報流通市場の拡大と情報システムの変化を背景にして，その共通な情報表現のルールとして XML に期待がかかり，データの XML 化が進展すると思われる[1]．したがって，XML データの高速解析のための環境が重要になってくる，と考えられる．

†1 株式会社セック SI 本部宇宙先端システムビジネスフィールド
Systems Engineering Consultants Co., Ltd
†2 株式会社セック マーケティング本部
Systems Engineering Consultants Co., Ltd

筆者らは，2002 年に，XML データベースを用い，XMLデータを扱ったデータマイニングの可能性を示した[2]．その後，上記で用いたインメモリ XML データベースを検索エンジンとして，宇宙航空研究開発機構宇宙科学研究本部（旧文部科学省宇宙科学研究所，以降 JAXA/ISAS と表記）の太陽観測データ解析環境実現のための，太陽観測(飛翔体・地上)統合データベース・公開システム向けのプラットフォームを開発した．この開発の中で，XML データに対して算術演算（四則演算，三角関数）を行った結果を使い XML を検索する必要が生じた．当初は，XML データベースから取得した検索結果に対して自前で計算して絞り込みを行っていたが，この方法では，コンピュータのバスネックとなり，データ取得量に応じて全体のスループットが落ちてしまう．そこで，XML データに対しユーザが定義した複雑な算術演算を行い，その結果をもとに検索する機能を実装して再評価を実施した．本論では，インメモリ XML データベースのシステムへの適用とその評価を通して，データ解析システムエンジンとしての XML データベース活用の可能性について考察する．

2. インメモリ XML データベース「Karearea」

2.1 Karearea の概要

XML の普及に伴い，従来の商用 RDB 製品が「ネイティブ XML 対応」を謳う一方で，XML データをそのままの形で取り扱うことが可能なネイティブ XML データベース(NXDB)製品が登場してきている[3] [4] [5] [6] [7].

しかし， XML の扱いが容易な半面，現状の NXDB は多量の XML データに対して十分な性能を出すことが比較的難しく，またインデックスを多用することによってメモリ消費量が多い，などの問題点もある．そこで，筆者らは大量の XML データから，インデックスなどを必要とせずに高速な検索や集計を行うことが可能なインメモリ XML データベース「Karearea（カレアレア）」を開発した[8].

Karearea は，株式会社ターボデータラボラトリーが開発した LFM（Linear Filter Method）技術[9]を基盤としており，全てのデータは主記憶上に置かれたうえで管理される．LFM では，FAST（Filter Array STructure）という効率的なデータ構造を採用することにより，主記憶上のデータに対して，インデックスを全く必要とせずに高速な検索，ソート，結合，多次元集計処理を実現している．また，独自のデータ圧縮機構を持ち，冗長度が高いデータを非常にコンパクトに格納することができる[10]．筆者らは，これら全ての機能を XML に対して行えるように拡張し，XML データベースエンジンとして製品化した．

2.2 XPath での複雑な検索条件指定

Karearea は，アプリケーションから渡された XML ドキュメントを解析，分解して LFM データベースに格納し，また XPath での検索指定に応じてデータベース内を検索して結果を XML データに組み立て直して返却する．（図 1）

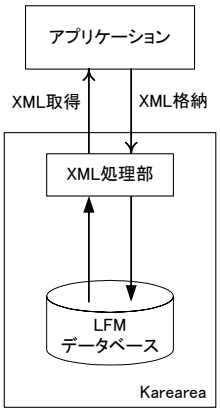

図 1 Karearea の構成

一般的な NXDB と同様に，Karearea でも，XML データの検索（絞り込み）のための条件指定に，W3C で制定された XPath 式[11]を使用する．図 2 の XPath は，株価データを表す XML データと，これらが大量にある中から銘柄コードが 1001 のものを検索する例である．

```
<stock-price>
  <trading-date year="2003" month="07" day="24"/
>
  <issue-data>
    <issue-category code='65'>通信</issue-category>
    <stock-index nikkei='0' topix='0'/>
    <issue-code>1001</issue-code>
    <issue-name>カレアレア</issue-name>
    <todays-start-price>1350</todays-start-price>
    <todays-high-price>1500</todays-high-price>
    <todays-low-price>1350</todays-low-price>
    <todays-end-price>1480</todays-end-price>
    <todays-output>45000</todays-output>
  </issue-data>
</stock-price>
```

```
/stock-price[issue-data/issue-code = 1001]
```

図 2 株価データと XPath の例

XPath 仕様では，要素の属性や，要素の子のテキスト値に対して加減乗除などの数値演算を行い，その結果での絞り込みを行うことも可能である．また，ビルトイン関数として小数点以下の切り上げや切り捨てを行うような数値関数も規定されている．

しかし実際のシステムにおいては，もっと複雑な演算や条件判定を行った結果を使って検索したいケースが少なくない．例えば，図 3 の XML データに対して以下のような三角関数を使用した検索を考える．

```
/root[(cos(rad) * q1 - sin(rad) * q2) < c ]
```

sin や cos といった三角関数は XPath のビルトイン関数としては規定されていないため，直接検索することはほぼ不可能である．（なお，rad，q1，q2，c は root の子要素である）

```
<root>
  <rad>1.04</rad>
  <q1>35.3</q1>
  <q2>54.8</q2>
  <c>10.0</c>
</root>
```

図 3　測定結果を表す XML データの例

そこで，代替案として XPath での検索を行うのではなく，いったんすべての root 要素を XML 形式で取得し，アプリケーションにて必要な値を取り出してから計算および絞り込みを行う方法が考えられる．例えば，この場合であれば各 XML について，rad，q1，q2，c 要素のテキスト値を取得して計算を行い，条件に一致するもののみを残すことになる．

Karearea では，検索結果セットに対して，レコードごとに残す，残さないを指定して，必要なレコードのみを残した新しい検索結果セットを返す機能が用意されている．この機能を利用することによって，アプリケーションでの絞り込みも容易に実現可能ではあるが，検索性能が大幅に劣ることは避けられない．また，使い勝手の面でも，あまりよいとは言えない．

あるいは，Karearea 独自のビルトイン関数として三角関数をサポートすることも考えられるが，アプリケーションごとに必要な関数は千差万別であり，すべてを用意することは不可能である．このような用途においては，例えば SQL 文における埋め込み関数呼び出しのように，自由にユーザが定義した関数を，XPath の他のビルトイン関数と同じように XPath 式で指定できることが望ましい．

一方，W3C で策定中の XQuery 仕様[12]でも「Function Declaration」として，ユーザが定義した関数や外部の関数を呼び出すことが可能である．また，外部関数を含んだ XPath 記述を Proximal Nodes モデルに変換して処理を行う方式も提案されている[13]．

しかし XQuery 仕様の方法では，独自の言語仕様に基づいてロジックを記述する必要があるほか，Java などのプログラム言語に比べると，実装できる機能は限られる．そこで筆者らは，XPath 式からユーザが Java 言語で記述した関数を呼び出せる「コールバック関数」の仕組みを考案し，Karearea に実装した．

2.3 検索指定条件文へのコールバック関数の導入

Karearea では，XPath 式の解釈に Apache プロジェクトで開発されたオープンソースの XSLT プロセッサ「Xalan」[14]を使用している．Xalan では，そのままでは XPath 1.0 仕様で定義されていない関数を扱うことができないが，Karearea でのコールバック関数の導入に際して Xalan の改造を行うことは避けたいと考えた．したがって，独自の記述方法を採用するのではなく，従来の文法に準拠した形で実現する方向で検討を行った．

XPath 仕様では，「$var」のような記述によって変数参照が可能である．変数名には XML の修飾名（Qualified Names）を指定することになっているが，Xalan では「$"abc"」や「$"a+3"」のように文字列を指定しても問題なく構文解釈されることを確認した．これを応用し，XPath 式に「$"func(X,Y)"」という記述をするとユーザ定義関数 func が呼び出される仕様とした．また，X や Y には数値や文字列，要素のテキストや属性だけでなく，計算式や他の関数呼び出しなども指定可能とした．

コールバック関数は，図 4 のように call メソッドを定義した Java のクラスとして記述する．call メソッドでは，Karearea の XPath 処理部から渡された引数（数値または文字列）に対して任意の演算を行い，結果を数値か文字列として返す．

コールバック関数による検索を行うには，関数名とクラスの対応をあらかじめデータベースに登録しておく必要がある．登録されたコールバック関数が XPath 式で指定されると，レコードごとに対応するコールバック関数クラスの call メソッドが呼び出される．call メソッドで値の評価を行い，結果が残りの XPath の評価に使用される．なお，コールバック関数は複数登録可能である．（図 5）

```
public class Func
        implements XPathUserFunction {
    public double call(double rad,
                       double q1, double q2) {
        return Math.cos(rad) * q1 -
            Math.sin(rad) * q2;
    }
}
```

図 4　コールバック関数

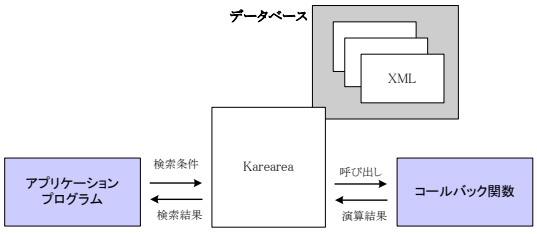

図 5　コールバック関数の実行

コールバック関数を導入することにより，結果として，前述の XPath は以下のように記述できる．

```
/root[$"Func(rad, q1, q2)" < c ]
```

コールバック関数には，単純な関数演算だけでなく複雑なロジックを記述することも可能である．したがって，原理的には XML データ内の値を使用した，いかなる検索条件でも指定できることになる. また, Java のクラスとして記述するため, コールバック関数で Java の豊富な API ライブラリを使用し，ファイルや他の RDB などにアクセスすることさえ可能である．したがって，単に複雑な検索を行うというだけでなく，使い方次第で様々な応用が考えられる．

3.　インメモリ XML データベースの適用

前記のような特徴をもつインメモリ XML データベース Karearea を JAXA/ISAS の SODA(SOlar Database & Archives)計画の統合データベース・公開システム向けのプラットフォームのプロトタイプシステムに適用した．SODA 計画は，現状の太陽観測の問題点を解決するために，国立天文台と JAXA/ISAS が中心となり，立ち上げている解析環境整備の計画である[15].

現状の太陽観測計画には以下のような問題点がある．

（1） 観測装置（観測衛星・地上望遠鏡）を横断したデータベースがない
（2） 特に検索システムが付随したものがない
（3） 観測データが各々の観測所，研究所に分散していて，一括管理されていない
（4） オフラインでしかアクセスできないものもある

上記問題点を解決するための解析環境の一貫としてインメモリ XML データベースを検索エンジンの核とした Web システムを構築した．システム構成は図 6 のとおりである．

本システムのユーザとなる研究者は, 任意のクライアント環境から Web ブラウザを介して，SODA システムのサーバに検索／集計処理やプラグイン処理の実行を指示（処理①⑤）する．サーバ環境では，Web サーバとして Apache[16]，Web アプリケーションサーバとして Tomcat[17]が動作しており, ユーザからの指示により，Java Servlets(Servlet)[18]もしくは Java Server Pages(JSP)[19]の技術を利用して開発された検索／集計処理（処理②）やプラグイン処理（処理⑥）が実行される．ここで，プラグイン処理とは，観測データのダウンロード処理やムービー表示処理など，ユーザが用意した任意の処理をプラグインとして本システムに組み込み，それを実行する機能（処理⑦）である．

SODA システムにおいては，対象とする観測データは, 観測装置毎にデータフォーマットが異なる. また，検索／集計するためにデータベースで管理する項目は，システム運用中に変更される可能性がある．したがって，本システムでは，観測データを RDB で管理するのではなく，RDB よりもデータ構造を柔軟に表現できる XML データで管理する方法を採用した．観測データは事前に XML データに変換し，Karearea に登録しておく．太陽観測衛星「ようこう」の XML データの例を図 7 に示す.

検索処理／集計処理のバックエンドでは，XML に変換した観測データが登録された Karearea が動作しており，ユーザに指定された検索条件を元に，Karearea の API を呼び出し, 観測データの検索／集計（処理③）を行っている．

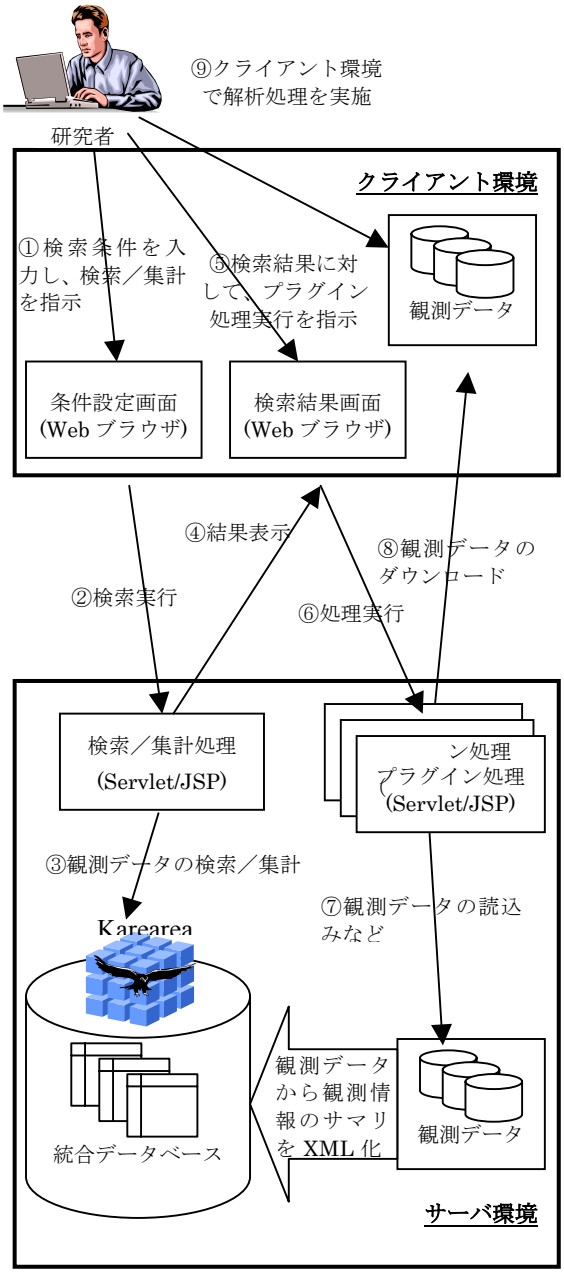

図６　SODA システム構成

```
<image public="true">
  <NAXIS1>1024</NAXIS1>
  <NAXIS2>512</NAXIS2>
  <DATE-OBS>1991-09-30T08:57:03</DATE-OBS>
  <CRPIX1>500.720</CRPIX1>
  <CRPIX2>563.990</CRPIX2>
  <CROTA>0.0955000</CROTA>
  <CDELT1>2.46</CDELT1>
  <CDELT2>2.46</CDELT2>
  <CRVAL1>0.0</CRVAL1>
  <CRVAL2>0.0</CRVAL2>
  <RADIUS>390.303724588</RADIUS>
  <DATAMAX>3436.00</DATAMAX>
  <DATAMIN>-64.0000</DATAMIN>
  <TELMTYPE>SFR</TELMTYPE>
  <TELMFILE>sfr910930.0804</TELMFILE>
  <FILTER-A>Open</FILTER-A>
  <FILTER-B>Al.1</FILTER-B>
  <RESOLUT>Full</RESOLUT>
  <EXP-TYPE>Dark</EXP-TYPE>
  <EXP-DUR>227909.</EXP-DUR>
  <EXP-LEV>36</EXP-LEV>
  <DP-MODE>Quiet</DP-MODE>
  <DP-RATE>High</DP-RATE>
  <file basename="sfr19910930_090411">
    <format>
      <type>-25.jpg</type>
      <size>40195</size>
    </format>
                    :
                (省略)
                    :
    <format>
      <type>.png</type>
      <size>264532</size>
    </format>
    <format>
      <type>.fits.gz</type>
      <size>546842</size>
    </format>
  </file>
</image>
```

図７　「ようこう」の XML データ例

4. SODA システムの性能評価

4.1 性能測定

SODA システムの性能要件は以下のとおりである．

（1） 複数ユーザ（2 名以上）による同時検索ができること

（2） 300 万件(画像ファイル 1 枚/分×約 5 年分)の XML データを登録／検索できること

（3） 日時，太陽緯度／経度，半径による検索が 5 秒以内に完了すること

（4） 10MB の画像データのダウンロードが 10 秒以内に完了すること

図 6 のシステムを用いて，太陽観測衛星「ようこう」の XML データを対象とし，以下のマシン環境で性能測定を行った．

Machine	Sun Blade 100
CPU	UltraSPARC-IIe 500MHz
Memory	1,152MB(128M*1, 512MB*2)
OS	Solaris8

性能測定は，総数 53,742 件と 107,671 件のデータを用意し，各々において，ヒット件数の推移と検索時間の関係について測定した．Karearea については，データ総数と検索時間の関係はほぼリニアである[2]が，今回の測定結果では，53,742 件と 107,671 件では，検索時間に大きな差は見られなかった．データ総数よりも，データヒット件数がより大きな因子として効いていることがその理由と考えられる．

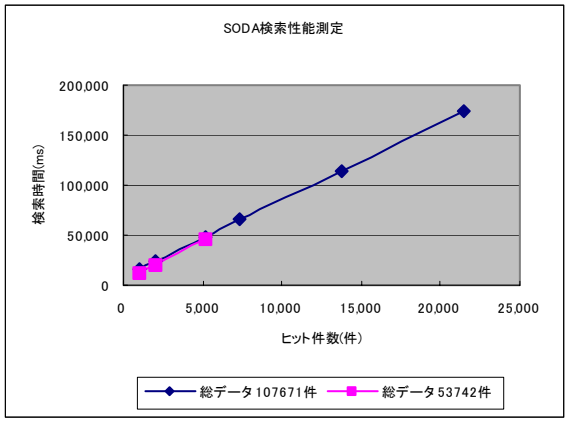

図８ 性能測定結果

4.2 性能測定の分析

性能測定の結果，ヒット件数が多くなれば多くなるほど，検索時間が長くかかっている．そこで，どこが検索のボトルネックになっているか，図 6 の処理③について，個別の処理時間を計測した．対象は，今回測定したヒット件数の最大値 21,526 件，検索時間 186,849msec のケースを抽出して分析した．結果は以下のとおりである．

（1） 検索条件(XPath)の作成：23msec

（2） 観測データの検索処理：731msec

（3） XML データの取得処理：176,668msec

（4） 表示のためのデータ集計処理：7,621msec

ここで，（4）は，Karearea の多次元集計機能を用い，（1）から（3）の一連の検索処理結果に対し，20 回程度の集計処理を再度走らせている時間である．

上記分析から，処理時間 186,849msec のうち，94%以上を Karearea からの XML データ取得に費やしていることが判明した．（4）が今回システムの画面表示のための特別処理であることを差し引き，純粋な検索の部分のみを抽出して考えると，XML データ取得の検索時間全体に占める割合は 98%を越える．検索時間そのものは 1 秒もかかっていない．データ取得がパフォーマンスのボトルネックとなっており，データの取得自体はバス性能および，XML データのサイズに依存することを考えると，バス上のデータ転送量が増えればある程度リニアに転送時間が増えることは避けられない．システム特性にもよるが，検索パフォーマンスの向上は，いかにデータ転送量を押さえて，事前に絞込みを行えるか，にかかってくる．

今回の性能評価に使用した「ようこう」の XML データは，Karearea に登録する前の XML ファイルのサイズは約 1.2KB／1 ファイルであるが，Karearea の LFM のデータ構造に変換して，メモリ上に格納した状態では約 1.8KB／1 ファイルとなる．53,742 件のメモリ上でのサイズは約 95MB，107,671 件では約 190MB になる．したがって，21,768 件の XML データの取得処理では，約 40MB のデータが LFM 上から本システムに転送されることになる．XML データ 1 件あたりの取得時間は約 8msec である．この取得時間が，件数の増大とともにリニアに検索時間に効いてくることになり，全体のスループットを悪化させている．

4.3 コールバック関数の活用

「4.2 性能測定の分析」にあるとおり，いくら検索性能が速くても，XML データの取得に時間がかかるのであれば，その検索性能は生きてこない．実際に大量にヒットした XML データの取得を必要とする処理の

場合はある程度避けられない時間としても，検索の絞込みのために一度データを取得し，それを元に演算をかけ，演算の結果で最終的なデータの絞込みを行う場合には，取得する XML データは最小限（結果のみ）としたい．ここでコールバック関数を活用することを考える．本システムでは，「日時，太陽緯度／経度，半径による検索」を行っているが，「太陽緯度／経度，半径」による検索は，一度 XML データを取得し，そのデータに算術演算をかけた結果を検索条件として生成する必要がある．すなわち，前述の計測結果で 21,526 件のデータに対して「太陽緯度／経度，半径」の情報をもとに最終的に 100 件に絞り込むケースにおいても，21,526 件のデータの取得が発生し，検索処理時間 186,849msec は低下できない．したがって，最終的な絞込み件数が 100 件でも 21,526 件でも有意な差は出ないことになる．しかし，ここにコールバック関数を適用すれば，不要な XML データの取得を避けることができる．

4.3.1　コールバック関数

本システムでは，観測データに付随する 9 種類のパラメタを使用し，加えて，ユーザが入力する 3 種類のパラメタを含めて太陽面上の緯度・経度を計算し，それを検索条件として設定する．本システムでは，この検索条件をコールバック関数で実現した．コールバック関数の呼び出し部分を図９に，コールバック関数の関数本体を図 10 に示す．

```
String xpath = "/image[$'SODAEvaluationFunction(
    CDELT1, CDELT2, "+ "CRPIX1, CRPIX2, "+
                "CRVAL1, CRVAL2, "
    +"NAXIS1, NAXIS2, " + "CROTA, " + radius + ",
                " + ew + ",
    " + ns + ")' = 1]";
```

図９　コールバック関数呼び出し部実装

図 10 からわかるとおり，コールバック関数では，Java プログラムで記述可能な算術演算であれば，あらゆる算術式を XML の検索条件として設定できる．また，その入力パラメタには，検索対象の XML のタグであれば何でも扱うことができ，外部入力値も扱える．

4.3.2　性能再測定

コールバック関数を使用して検索条件を設定し，性能の再測定を行った．データ総数 107,671 件に対し，1,056 件のデータを抽出し，その中から 1 件のデータに絞り込むケースを測定した．測定結果を表１に示す．

```
double r = radius / Math.abs(cdelt1);

double cosa = Math.cos(crota);
double sina = Math.sin(crota);

double q1 = (ew - crval1) / cdelt1;
double q2 = (ns - crval2) / cdelt2;

double p1 = cosa * q1 - sina * q2 + crpix1;
double p2 = sina * q1 + cosa * q2 + crpix2;

boolean included = (p1 - r >= 0) && (p1 + r <
                naxis1) &&
    (p2 - r >= 0) && (p2 + r <  naxis2);
```

図 10　コールバック関数計算部実装

表１　コールバック関数使用による絞込み効果その１

コールバック関数未使用	コールバック関数使用
13,545msec	5,569msec

上記のとおり，コールバック関数未使用時と使用時では検索時間が半分以下に短縮されていることがわかる．ただし，上記の結果は，今回システムの特徴的な処理（「4.2　性能測定の分析」（４）の処理）を含んでいるため，より純粋な検索処理のパフォーマンスの向上具合を調べた．当該処理を省き，純粋な検索のみの箇所を抽出して測定した結果を表２に示す．

表２　コールバック関数使用による絞込み効果その２

コールバック関数未使用	コールバック関数使用
9,038msec	814msec

表２から明らかなとおり，コールバック関数による事前の絞込みにより，純粋な検索処理のスループットタイムは 1/10 以下となることがわかった．すなわち，XMLデータを転送するためのバスのボトルネックは完全に解消することができたのである．

5.　今後の課題と展望

我々は，インメモリ XML データベースの Karearea を開発し，Karearea を実システムに適用した．実システムへの適用に際し，XML の要素を抽出して演算した結果をもとに絞込み検索を行う必要性が生じたが，これは，絞込みのための XML データの取得が処理全体のスループットを悪化させることになり，データのバス転送の壁にぶつかった．そこで，XPath の仕様に準

拠したまま，XPath でユーザ定義のコールバック関数を呼び出す仕組みを考案し，実装した．また，このコールバック関数を，実システムに適用することで，その有効性を検証した．市販パッケージ製品である Karearea も，それ単体では速さ，使い勝手，有効性などを本当の意味では評価できない．本論にあるように実システムに適用することにより，初めて気付くこともあり，改善の機会を得る．今回のコールバック関数の実装と適用はその好例である．Karearea は，今後も今回のような実システムへの適用を通して，さらに使い勝手の良い，インメモリ XML データベース製品として改良を重ねていくことになる．

また，今回適用したコールバック関数は，データベース上でのデータの絞込みには非常に効力を発揮する．Karearea のコールバック関数は，データをすべて取得し，手元であらためて作業する場合には有効ではないが，何らかの知見が必要で，大量データの中から，必要な情報だけをピックアップしてくる作業に関しては，あらゆる業務に有効に適用できると考えられる．もしかすると，前者に対しても，工夫次第では人手の負荷軽減になる可能性もある．それは手元であらためて作業することは人間の判断が必要なためであり，コールバック関数の持つインテリジェンスが，人間の試行錯誤のある程度の手助けになるだろう，と考えられるからである．そのような可能性を考えられるほど，このコールバック関数の記述は柔軟であり，XML データ内の要素のテキスト値，属性値を扱った，複雑な計算を伴う検索条件が指定可能である．これは，例えば，科学技術分野で XML が活用されるにあたり，複雑な計算を行う検索や統計解析にも，大いに役立つと考えられる他，XML の普及と伴に活用範囲は広がっていく，と考えられる．

我々はまだ入り口に立っただけであるが，データ解析エンジンとして Kareara を適用できる目処は立った．今後は，このコールバック関数の応用，適用の可能性をさまざまな解析の現場に適用していき，その可能性を探っていきたい，と考えている．それとともに Kareara の機能向上を図っていくことになる．

謝辞　本稿で述べたコールバック関数のアイデアは，JAXA/ISASの松崎恵一氏にアドバイスいただきました．また，SODA計画のシステム化全般にあたり，数多くの有益な意見をいただきました．

参 考 文 献

1）XML フォーマットのイエローページ「XML とは」
http://it.jeita.or.jp/eltech/XML/index.html
2）長瀬雅之，森田康裕，中本啓之，飯田　学，小林祐介，土井憲雄: XML データベースを使用したデータマイニングの試み，日本ソフトウェア科学会　データマイニング研究会　第3回データマイニングワークショップ, ISSN 1341-870X No.23, pp.21-24 (2002).
3）土井憲雄: LFM を使った高速インメモリ XML データベース Karearea，日本データベース協会 データベース No.21, ISSN 0288-5662, pp.24-27 (2002).
4）Tamino XML Server
http://www.softwareag.com/tamino/
5）Sonic XML Server
http://www.sonicsoftware.com/
6）Yggdrasill
http://www.mediafusion.co.jp/
7）NeoCore XML Management System
http://www.neocore.com/
8）川口浩司，沼崎慎吾，柳下精一郎，土井憲雄: インメモリ XML データベースの開発，情報処理学会研究報告　Vol.2003，No.78，ISSN 0919-6072，pp.9-16 (2003).
9）古庄晋二: 大規模データの超高速処理技術，日本データベース協会 データベース No.21，ISSN 0288-5662, pp.20-23 (2002).
10）Developer ヒーロー 導入事例 4 【株式会社ターボデータラボラトリー】
http://sdc.sun.co.jp/developers/community/person/2003/hero2003_002.html
11）XML Path Language (XPath) Version 1.0
http://www.w3.org/TR/xpath
12）XQuery 1.0: An XML Query Language
http://www.w3.org/TR/xquery/
13）永井孝明，石川佳治，北川博之: 外部関数を含む XML パス表現に対する問合せ処理手法，電子情報通信学会データ工学研究会 第 13 回データ工学ワークショップ(DEWS2002)論文集，ISSN 1347-4413，A6-2 (2002)
14）Apache Xalan
http://xml.apache.org/xalan-j/index.html
15）宇宙航空研究開発機構　宇宙科学研究本部　宇宙科学情報解析センター: SODA(Solar Database & Archives)計画，PLAIN センターニュース，第 109 号 (2002).
16）The Apache Software Foundation
http://www.apache.org/
17）Apache Tomcat
http://jakarta.apache.org/tomcat/index.html
18）Java Servlet Technology
http://java.sun.com/products/servlet/
19）Java Server Pages Technology
http://java.sun.com/products/jsp/

（平成 15 年 9 月 29 日受付）
（平成 15 年 10 月 27 日採録）