デジタル・ドキュメント 34－3
（２００２．７．２６）

XMLデータベースの種別と活用法上の留意点および問合せ言語への要求について

野村直之*1　川崎洋治*2
***1 法政大学エクステンションカレッジ　*(株)ジャストシステム**

概要：　XML ベースの技術が、B2B, B2C、そして、基幹系(数値系)と情報系(文書・マルチメディア系)を融合し、出自の異なる異種の情報を統合してビジネスを自動化し、また広域情報提供、学術研究の枠組みを変えつつある。本論文ではまず、これらの応用の高度化の鍵を握る XML データベースの３種の基本構成、そして、各構成ごとの何種類かの XML 文書の格納方式について論じる。さらに、これらの構成、方式から示唆される性能、計算資源利用効率上の各特徴に触れた後、管理機能や付加機能(例：別な構造の XML 文書の形で取り出す) について論じる。特に重要な問い合わせ言語について、人間向けの標準 XQuery と、機械向けの標準 XQueryX に分化しつつある現状を紹介する。最後に、従来の RDB, ODB に欠落している XML 固有のデータモデルに起因する留意点を指摘し、今後求められる XML データベース・エンジンの進化の方向性や利用形態の変遷について私見を提示する。

A Note on XML Database, its Deployment Strategy, and the Requirements of XML Database Query Languages

Naoyuki NOMURA*1　Yoji KAWASAKI*2
***1 Hosei University Extention College　*2 Justsystem Inc.**

Abstract:　We present a classification and the comparison among some architecture for XML Database.　Some merits and demerits of XML databases are discussed so that the users can decide on which kind of XML database to deploy.　We compare to SQL the emerging standard for XML database query language, namely XQuery, which is also compared with XQueryX that is supposed to function as the better machine understandable XML database query language.

1 はじめに　～今なぜ XML データベースか？

Web の発展、ブロードバンド化を見越した e ビジネスは、次世代 Web インフラの構築素材 XML を蓄積し活用する XML データベースへの莫大な潜在ニーズを生んでいる。特に、eCRM (electronic Customer Relationship Management) の先鋭的な思想の１つ OneToOne マーケティングでは、個々の顧客の行動履歴、対話履歴、プロファイル等から推定される潜在ニーズを構造化して可能な限り多量に蓄積し、次のニーズを精密に予測しようとする[野村 2001]。我々は、その実現のためには、洗練された XML データベース活用技術が不可欠と考える。

データベース・マーケティングは、かつて特殊な先端ＩＴ応用分野として経営学方面で研究されていた（例えば[Jackson94]）。しかし今や、Web を用いた BtoC, BtoBtoC, eCRM, OneToOne 等のキーワードに象徴されるマーケティング手法として、極めて普遍的、一般的なものとなっている。一般に不定形の、様々な構造（深浅、長短、再帰の有無等）、寿命、データならびにスキーマの更新頻度をもつ XML 文書を如何に安全確実に蓄積し、高速に、「知的に」活用できるか。また、SCM (Supply Chain Management) と総称される、購買・生産・物流・価格形成・決済の一連の流れにおける各モジュールの最適化にも、XML が使用され始めている(例えば[西岡 2001][野村 2001])。これらの応用が進むにつれ、プライバシ確保、著作権管理、アクセス権管理、部分暗号化、高度な分散処理に対応した Web アプリケーションサーバとの連携、等の新しい周辺機能、管理機能が求められるようになった。XML データベースに特徴的な新しい要求として、構造変換、言語変換、マージ（自動編集）しながら検索結果を取り出す、などが出てきている。

XML データベースへの期待の１つは、伝統的に、基幹系，情報系に分断されていた情報システムを統合することである。銀行のオンライン勘定系に代表される基幹システムでは、スキーマは原則不変であり、データの殆どは数値情報である。これに対し、その他の様々な情報を扱う情報系では、多様なスキーマが混在しているか、もしくは実質的にスキーマの無い、テキストや html、独自バイナリー形式の文書ファイルや図表データを扱う。XML は両者を統合することができる。

XML データベースの主な応用領域は、次の３つに大別できる：

(1)大規模 WebSite の構築・管理

(2)文書管理・知識管理　　（イントラネット、エクストラネット）

(3)基幹系と情報系の統合　（数値と自然言語）（データウェアハウス）

(1)についていえば、XML データベースなくしては大規模 Web サイトの管理、運営は事実上不可能な状況になっている。実時間で更新される顧客情報、決済情報、製品情報、在庫情報、物流情報、これらの履歴管理、視覚化、等々、すべて Web 上の分散コンピューティングでこなしつつ、データは安全に一元管理されなければならない。Web 上で GUI を提供するからには、クライアント端末上では HTML か XHTML に変換されるのが必須である。この画面上で入出力される情報と、各種データベースが相互依存関係にあり、スキーマを含めて常時変化する（図 1)。

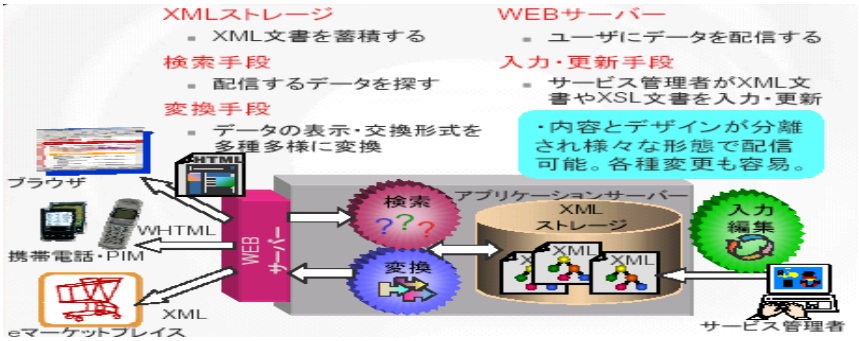

図 1　大規模、且つ動的な Web system の典型的な構成

(法政大エクステンションカレッジ 2001 年秋期講座『新ビジネスモデルと新言語設計』 林浩一講師分から引用)

競争力のある WebSite を構築し発展させるためには、その１カ所でサービスが完結する "One Stop Service" を指向した企画と、それを支える技術の裏付けが必要である。そこで、オープンで安価で汎用性の高い XML 関連技術が必然の選択肢となった。2000 年９月以来、WebService と総称される XML ベースの Web 上のアプリ統合技術体系が急速に注目を集め、開発、試行されるようになった。WebService の場合も、大規模で頑健な仮想サイトによって差別化をはかるならば、各種 XML データベースを分散サーバ上の各所に適切に配置し、適切なスケジューリングによりデータの連携を行うのが必須と考えられる。

以下、第２節では、まず XML データベース・エンジンのコアの違いによる基本構成の種別について論じる。さらに、いわゆる 'ネイティブ'XML データベースと呼ばれるエンジンの典型的な原理、内部実装方式を紹介する。第３節では、数種類の RDB への XML 文書の格納方式の違いを示し、その得失、さらには、RDB 利用そのものに共通する得失について論じる。第４節では、XML-DB 問合せ言語とその使い分けについて論じる。標準化が進みつつある標準化が進みつつある XQuery を紹介し、SQL の基本演算のいくつかが XQuery でどう実現できるかを示す。さらに、ヒトのための XQuery と並行して利用されると予想される、機械可読な XQueryX を紹介する。第５節で、RDB, ODB と対比した XML-DB のデータモデルを示し、XML-DB 特有の周辺機能、管理機能の今後の発展を予想する。

2 XML データベースの基本構成の種別 と 'ネイティブ'XML データベース

XML の登場が比較的最近で、且つ、XML を出し入れして一元管理できるデータベースへの需要が切実であったことから、既存の３種類のエンジンを XML 入出力に対応させたものが使われている。それは、関係データベース RDB、オブジェクト・データベース ODB、そして、全文検索用途のテキスト・インデクシング・エンジンに独自のファイル構造、メモリ管理を付加したもの、の３種類である。商品名として、「XML ネイティブデータベース」という名が冠されるのは、RDB 利用以外の２種類についてである。

オブジェクト・データベース ODB を元にした XML データベースは通常、DOM (Document Object Model) と呼ばれる、メモリ上で XML 文書の各ノードを高速にアクセスできるデータ構造を忠実に再現する。すなわち、ODB 内部のオブジェクト間をリンクして木構造にしたストレージを構成し、あとは ODB 本来のメモリ管理や更新履歴等の機能に委ねる、というやり方である。これにより、ユーザ、すなわちアプリケーション構築者からみると、XML 文書を入力、更新、出力する、という操作を考えるだけで良いようになっている[1]。ODB を元にした XML データベースの代表的製品には、eXceLon 社の XIS (eXtensible Information Server)がある。

ODB の内部コアに DOM をマッピングした方式では、一般に、任意のノード群（部分木）を再帰的に高速アクセス可能となる代償に、インデックス量が大きめとなる。しかし、実際の製品では、DOM のメモリ消費量の目安、すなわちドキュメント・ソースの８倍から 10 数倍というサイズに比べれば著しく小さい、２～３倍程度以下に抑えられている。にも関わらず、１文書が 100MB 以下であれば、DOM の高速ノード・アクセスに準じた性能特性が得られている。

テキスト・インデクシング・エンジンを XML の文書構造に対応させ、XML 文書中のノードへのアクセスの効率をはかった方式のものは、狭義の「XML ネイティブデータベース」と呼ばれる場合がある。しかし、製品や使用法によっては DOM の API 操作のすべてがサポートされていない場合もあり、注意が必要である。木構造を効果的に活かせる独自のインデクシングを採用しているだけに、速度やメモリ消費量の点で、原理的に最適化の余地が大きい方式となっている[2]。この方

---

[1] より高次の API や問合せ言語、マクロなどのミドルウェア機能による差別化もはかられている。

[2] 但し、ODB の方も、頻繁に通る Path、ノードを考慮してメモリ管理を最適化するようにチューニングを施す余地は存在する。その採否は技術的要因というよりも市場、開発コスト如何に関わっている。

式の代表的製品には、Yggdrasill (MediaFusion 社)、Tamino (独 Software AG 社,BeaconIT)、Zelkova[3] (Infoteria 社)、Karearea((株)セック)がある。

テキスト・インデクシング方式のインデクシング時間、すなわち初期データベース構築（格納）に要する時間は、同一製品であっても、インデックス対象のノード、インデックス構造、入力 XML 文書の構造・一様性によって異なってくる。また、検索性能については、大量のメモリを実装してインデックス全体をオン・メモリ動作させれば非常に高速になるが、大規模データベースでは通常部分的なオン・メモリ動作となる。このため、所要性能に応じた最適化を行うことによって対費用効果を確保できるようになる。

RDB ベースのものも含めて、各種速度は、OS のファイルシステムやメモリ管理に依存する部分がある。データベース・エンジンによっては OS の制御を実質的に奪ってしまうものもあるが、そうであっても入出力の際に OS や他のミドルウェアに何らかの影響を受けるため、性能評価には十分な注意が必要である。

3 RDB への XML 文書の格納

前節までで、RDB ベースの XML データベースは比較的単純で固定的な繰り返し構造のデータの格納に向くようなことを示唆したが、RDB への XML 文書の格納や運用の方式の違いによって、必ずしもそうではない。多くの RDB 製品がサポートする BLOB (Binary Large Object)や CLOB (Character Large Object)と呼ばれる不定形データ格納領域に不定形の XML 文書を１文書丸ごと格納するやり方があるからである。しかし、このやり方では、関係代数に裏書きされた高度な操作や、構造化検索機能、高速性を発揮させることは期待できない（但しベタ書きテキストとして BLOB 内を全文検索することは可能）。

そこで、XML 文書中の要素全てまたは一部を RDB のスキーマにマッピングし、データベースへの入力の際に XML 文書をパーズ、出力の際にシリアライズ[4]する、という方式が考えられ、普及している。また、RDB の表中の１要素として XML 文書を位置づけ、その内部構造をユーザ定義型で与え、検索可能にしておく、という方式も使われている[5]。

以上の３方式を図解したのが図２である。

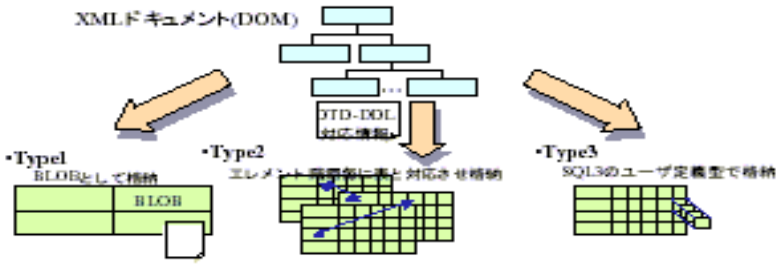

図2 RDB に XML 文書を格納する主要な3方式

(JavaConsortium XML 部会資料:NTT コムウェア山田洋一著『既存 RDB の XML 対応調査報告』より引用)

Oracle 9i (Oracle 社)、DB2(IBM 社) 等の商用 RDB 製品は、図２の Type1,Type2, Type3 の全部または一部を、何らかの方法で使えるようにサポートしている。例えば、Oracle 9i の Type1 サポートは、CLOB を拡張して XML 型を定義し、アクセス API を提供している。この部分をとってみれば「XML ネイティブデータベース」と称するのも妥当でないとはいえない。

---

[3] Zelkova はタグの量、比率が小さいプレーンテキスト同然の XML 文書であっても、テキスト部分を形態素解析してインデクシング可能である。全文検索とスケーラブルに融合したユニークな製品。

[4] テキストデータとして実体化し、「書き出す」こと。

[5] この他、差分管理に適した、XML の要素を RDB の１レコードに展開するという方式が提案されている。http://www.xmlconsortium.org/member/public/seminar/w02/20020206_03.pdf

３方式の使い分けにあたっては、まず、完成イメージを描き、どのような動的な XML の活用形態をとるのかの全体図を描くべきである。ある要素の内容や属性値を鍵（キー）に高速検索を行う必要があるのか、スキーマの更新頻度はどうか、その際にデータベースの動作を停止できるか等、具体的に要求仕様、要求性能を見積もることが望ましい。その上で選択肢が残り、RDB の中でも複数の実現方式が考えられたならば、既存資産や現在の運用状況を見積もるのが効果的である。すなわち、XML データベース利用開始の時点で、既存の RDB やその他の文書ファイル等の資産が、どんなものがどの程度の分量存在するのか見極め、開発ならびに運用面の移行コストを見積もるのが望ましい。一般的傾向としては、Type1，Type3，Type2 の順に独自の高機能化、高性能化の余地が高くなる反面、開発コストが高くなる傾向にある。

RDB のスキーマへのマッピングは、Type1，Type3 の場合は、原則的に、RDB 製品のサポートする独自機能によるしかない。Type2 の場合、ユーザが独自のアプリケーションを組んで変換する方法(Level1)、ミドルウェアで対応する方法(Level2)、DBMS 製品の独自機能により DB と一体化し、XML ネイティブデータベースに近い扱いをする方法(Level3)、の３通りが存在する。対費用効果については一概に言えないが、初期開発・改造の工数の観点では、Level1,Level2，Level3 の順に優位になる。汎用性、特に RDB の移行性についていえば全く逆になる。これらの基本的な考察条件とともに格納対象となる XML 文書自体の構造がどの程度固定的であるか、その平均サイズはどの位か、などの条件を十分に分析し、実験した上で実装方法を選定するのが望ましい。

性能、すなわち、データベース構築や XML 文書内部のノードアクセス時の速度やメモリ消費量については、同一の製品であっても、上述のタイプの違い、スキーマへのマッピング方法の違い、そして、XML 文書自体の構造の違いによって大きく変わってくる。

4 XML-DB 問合せ言語とその使い分けについて

XML-DB コアや、DB 入出力･更新管理 API を構成するミドルウェアについて、様々な実用方式が存在することを前節までにみた。本節では、これらの DB コア、ミドルウェア実装の違いを超えて、XML 文書の入出力と更新制御を記述する共通言語としての XML-DB 問合せ言語について論じる。

4-1 XQuery について

```
<children>
    <child  birth='1995/12/5' sex='male'>
        <name>田中大輔</name>
        <favorite>
            <name>りんご</name>
            <name>ケーキ</name>
        </favorite>
    </child>
    <child  birth='1997/1/24' sex='female'>
        <name>荒木麻衣</name>
        <favorite>
            <name>いちご</name>
            <name>プリン</name>
        </favorite>
    </child>
    <child  birth='1996/6/15' sex='male'>
        <name>佐藤涼太</name>
        <favorite>
            <name>いちご</name>
        </favorite>
    </child>
</children>
```

図3-1 XML文書例 (children.xml)

```
for $c in
    document("children.xml")//child
      let $cn:=$p/name/text()
      where $c/@birth>="1996.1.1"
      return <child>{$cn}</child>
```

図 3-2 XQuery: 96 年以降生れの子供の一覧

```
<?xml version="1.0" encoding="UTF8">
<xql:result
xmlns:xql="http://metalab.unc.edu/xql/">
    <child>田中大輔</child>
    <child>荒木麻衣</child>
    <child>佐藤涼太</child>
</xql:result>
```

図 3-3 上の XQuery を children.xml にかけた結果

異種の情報源の統合という XML の役割を考えると、仮に複数種類の XML データベースを同時に使い分けていたとしても、それらにアクセスする問合せ言語は１種類に統一したい。この要求は自然である。また将来、対象データのスキーマやデータサイズが桁違いに拡大して、データベースエンジンを取り替えることになっても、問合せ言語で書いた過去のアプリケーションはそのまま使い続けたいい。そのためにも問い合わせ言語の統一が望ましい。

問い合わせ言語のシンタックス（書式、文法)、機能仕様については、比較的最近まで製品ごとにばらばらに、独自のものが使われていた。しかし、まだドラフト段階ながら W3C が XQuery をとりまとめ、推奨していることから、XQuery が事実上の標準となっていく可能性が高い[XQuery2002]。XQuery は、For 節,Let 節,Where 節,Return の頭文字をとった FLWR 式を基本にしている。問い合わせ言語自体は XML1.0 のシンタックスに準拠していない[6]。図 3-1 children.xml に対して、「1996 年以降に生まれた子供の一覧」を問い合わせる XQuery が図 3-2 である。問合せ結果は XQuery 独自の名前空間(接頭辞 xql:) を用いて、図 3-3 のように生成される。

4-2 SQL の基本演算を XQuery で表す

RDB の標準問合せ言語 SQL は、関係代数という数学モデルに基づいて、和(union), 共通(intersection),差(difference), 射影(projection),選択(restriction),直積(product),結合(join),割算(division)をもっている。これらのうち一見 XML-DB では複雑、困難そうな演算、射影、選択、直積、結合を、XQuery は比較的単純に自然に表現することができる。例えば「射影」は、必要なノードや属性を Path で抜き出すだけである(例: //child/name)。「選択」は、Path のフィルタ機能で絞り込むか(例: //child)、FLWR 式の where 句で絞り込む(例: where $c/@birth>=”1996/1/1”)。
「直積」は、ＦＬＷＲ式のｆｏｒ句で実現できる。一般の「繰り返し機能」を用いて実現できる。

（例）子供と、重複を除いた好物の可能な組み合わせすべてについて繰り返す

```
let $d:=document(”children.xml”)  for $c in $d//child,  $f in distinct($d//favorite/name)
```

「結合」は、「直積＋選択」であり、FLWR の for 句で直積、where 句で選択、により実現できる。

4-3 XQuery vs. XQueryX　～ヒト vs. 機械のための XML-DB 問合せ言語の分化

第 4 節冒頭に記した問合せ言語の要件を満たす言語は２種類以上あってもよい。違いに等価両者が双方向に容易に一意に自動変換できるならば、１つは人間可読な標準、もう１つは機械処理用の標準にしてはどうか。恐らくこのように考えた W3C (World Wide Web) のメンバーは、XQuery を XML のシンタックスで表現した XQueryX を提案した[XQueryX2002]。XQueryX は、XML パーザを共用可(引数となる部分 XML 文書のパーザと共用)でき、また、XQueryX で書いた問合せ文ライブラリを対象に検索できるなど(文書”**”を検索対象としている問合せ文を探す)、機械処理上、利点が多い。そして、雛形を加工して問合せ文を自動生成するにも、XML 出力の基本ライブラリが流用できて便利である。また、ＸＭＬ文書内に検索条件を埋め込め、埋め込まれた条件で当該文書自身から情報抽出できるなど多大なメリットがある。しかし一般に、非常に長くなり、且つ、XQueryX の部分と、引数内容の XML 部分記述が目視で区別しにくいため、人目に理解するのは極めて困難である。図４の書籍データベース bib.xml から名前と平均価格を返す XQuery 問合せ文を XQueryX で記述したもの(付録)と対比して確認されたい。

```
for $p in distinct(document(”bib.xml”))//publisher
let $a:=avg(document(”bib.xml”)//book[publisher=$p]/price)
return
    <publisher>
        <name>{$p/text()}</name>
        <avgprice>{$a}</avgprice>
    </publisher>
```

図 4 書籍 DB bib.xml への XQuery 問合せ文

[6] その理由は、引数やテンプレートとして頻繁に XML 文書の一部が記述されるため、問い合わせ言語自体が XML 化されると繁雑で視覚的に識別困難になるため、と推察される。

このように等価な XQuery と XQueryX とを共存させていくのであれば、相互に自動変換するライブラリが、あらゆる XML-DB 利用環境に配備されてほしい。そうすれば、人手で作成した問合せ文の改修を機械が引き継ぐのはもちろん、逆に、機械が自動編集・修正した XQueryX を可読性の高い XQeury に変換し、人間が創造的な改良（速度、機能）を引き継ぐことができる。

5 データモデル、その他の検討課題

XML が得意とする多様な情報源の統合利用環境においてデータベース・アクセスの一元管理のメリットを発揮するためには、データモデルの設計という重要課題が残されている。図５に、RDB, OODB, XML のデータモデルの比較を示す。

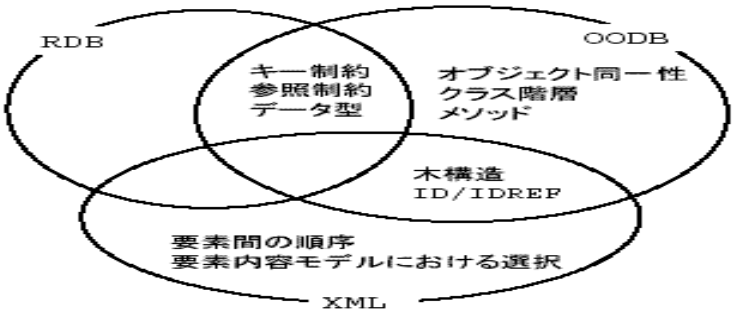

図5 RDB、OODB、XML のデータモデルの比較　～[吉川 2002]より引用

３節までの議論では、実際の XML 文書、実装された各種エンジン、アクセス方式の大別により、用途に応じた XML データベースの選定基準を洗い出すのに務めた。図５は、原理的な、情報構造の表現可能性の過不足を図示している。ここから読みとれることの１つは、XML データベースは必ずしも RDB, ODB(OODB)の能力を全て活用していない、という点である。それ以上に懸念されるのは、逆に、XML のデータモデルの中で、RDB, ODB では表現できないものがある点であろう。

この内、「要素間の順序(兄弟の序列)」については、確かに XML1.0 の規格の中で、保存されねばならないとされている。何の明示的な指標や構造をもたずにただ並べただけの順序情報を保証しろ、ということである。テキスト・インデクシング方式の場合は、ファイル先頭から後方へとシリアルに処理し、先頭位置からのアドレスを元にインデックスを付与する。従って、特に何もせずとも、データベースの構築、更新の全過程を通じて要素間の順序情報は保存される。これに対し、RDB や ODB においては原理的にはその保証を与えられない。

そこで、必要に応じて、XML 対応させるエンジン側で内部的にその保証ができるように拡張することになる。あるいは、ユーザ側が独自にアプリケーションを構築する場合には、兄弟要素間の順序情報を属性値等の形で明示的に表現し、保存する必要が生じる。いずれの場合も、文書の追加、削除等の更新管理の際にこれらの数値を更新するオーバーヘッドが発生する。また、XML 文書を取り出す際に、ソーティングのアルゴリズム相当の処理時間[7]がかかるようになる。

この他、XML データベース選定の参考基準となる付加的機能を、以下順不同に列挙する。

- XML スキーマ管理支援機能、
- XSLT による言語変換、情報抽出の支援機能、
- 接続性拡大のためのミドルウェア、
- Web サービス対応機能、
- WebSite における DB アクセス・ボトルネックの解決のための動的キャッシュ技術、
- アプリケーション、コンポーネントとの Data binding、
- スタイル、リンクをサポートするミドルウェア、
- 部分暗号化・解読をはじめとするセキュリティ機能、
- XML1.0 の規定外の文字コード、外字のサポート機能、

[7] 一般に、比較対象となる文書数 N に対し O(log(N)) の処理時間が余分にかかる。

これらの問題解決のための支援機能を適時に、必要十分に、使いやすく提供するための差別化競争が、XML データベースのベンダー間で行われている。

6　おわりに

以上、XML データベースの必要性(利用価値)、方式分類、各方式の性能、機能上の特徴と、使い分けのための留意事項について、私見を交えて述べた。また、問合せ言語への新しい要求、標準規格 XQuery の SQL からの移行性、機械理解用の XML ベース問合せ言語 XQueryX について論じた。XML ならではの新しい要求機能、性能、管理機能も重要であり、これらは XML 個別言語自体、またスキーマ設計、さらに、データ生成モデルと深く関わりつつ、システムの能力を規定していく。XML の多彩な用途からしても、万能の１つの XML-DB エンジンは無いと考えるべきである。利用目的、利用条件・環境をよく吟味し、データ設計、シミュレーションを十分に行った上で、用途に即した性能レンジ、拡張性(XML の 'X' (eXtensibile))、問い合わせ言語のサポート状況、移行容易性、等を判断し、一般には複数のエンジンを組み合わせ、使い分けていく必要があるだろう。

参考文献

[Jackson94] Rob Jackson and Paul Wang: “Strategic Database Marketing”, 1994, NTC Business Books, 1994
[野村 2000] 野村直之:『1to1 マーケティングに基づく E－ビジネスのための技術要件』, 情報処理学会第 25 回デジタルドキュメント研究会予稿集, 2000
[川崎 2002] 川崎洋治:『構造化検索とＸＭＬデータベース検索技術』, 法政大エクステンションカレッジ 講座『ナレッジマネジメントと XML』テキスト, 2002.5
[西岡 2001] 西岡靖浩編: “生産管理・スケジューリング言語 PSLX”, 2001
[野村 2001] 野村直之: “ ”, 2001
[野村 2002a] 野村直之:『2.1 XML データベース概説』,前川徹編“デジタルネット時代における情報化戦略調査委員会報告書”, (財)データベース振興センター, 2002.3
[野村 2002b] 野村直之:『ナレッジマネジメントと XML』, http://xml.fujitsu.com , 2002.7
[XQuery2002]“XML Query Requirements”,W3C WD, 15 Feb. 2001, http://www.w3.org/TR/xmlquery-req
[XQueryX2002] XQueryX”,W3C Working Draft, 15 Feb. 2001,http://www.w3.org/TR/xqueryx
[吉川 2002]吉川正俊:『データベースの観点から見た XML の研究』,情報学シンポジウム 2002, http://research.nii.ac.jp/sigfi/sympo/2002/proceedings/invited/yoshikawa.pdf

付録　図 4 の XQuery と同等の XQueryX（書籍 DB bib.xml への XQueryX 問合せ文）

```
<q:query xmlns:q="http://www.w3.org/2001/06/xqueryx">
  <q:flwr>
    <q:forAssignment variable="$p">
      <q:function name="distinct">
        <q:step axis="SLASHSLASH">
          <q:function name="document">
            <q:constant datatype="CHARSTRING">bib.xml</q:constant>
          </q:function>
          <q:identifier>publisher</q:identifier>
        </q:step>
      </q:function>
    </q:forAssignment>
    <q:letAssignment variable="$a">
      <q:function name="avg">
        <q:step axis="CHILD">
          <q:function name="document">
            <q:constant datatype="CHARSTRING"> bib.xml</q:constant>
          </q:function>
          <q:step axis="CHILD">
            <q:predicatedExpr>
              <q:identifier>book</q:identifier>
              <q:predicate>
                <q:function name="EQUALS">
                  <q:identifier>publisher</q:identifier>
                  <q:variable>$p</q:variable>
                </q:function>
              </q:predicate>
            </q:predicatedExpr>
            <q:identifier>price</q:identifier>
          </q:step>
        </q:step>
      </q:function>
    </q:letAssignment>
  <q:return>
<q:elementConstructor>
      <q:tagName>
        <q:identifier>publisher</q:identifier>
      </q:tagName>
      <q:elementConstructor>
        <q:tagName>
          <q:identifier>name</q:identifier>
        </q:tagName>
        <q:step axis="CHILD">
          <q:variable>$p</q:variable>
          <q:nodeKindTest kind="TEXT" />
        </q:step>
      </q:elementConstructor>
      <q:elementConstructor>
        <q:tagName>
          <q:identifier>avgprice</q:identifier>
        </q:tagName>
        <q:variable>$a</q:variable>
      </q:elementConstructor>
    </q:elementConstructor>
  </q:return>
</q:flwr>
</q:query>
```