【出展】
著者名：山田祥寛
ウェブページタイトル：XMLデータベース製品カタログ2003 ～ネイティブXMLデータベース編～
URL：https://www.atmarkit.co.jp/fxml/tanpatsu/28xmldbcatalog/nxdb01.html
https://www.atmarkit.co.jp/fxml/tanpatsu/28xmldbcatalog/nxdb02.html
最終アクセス年月日：2003年9月26日


# XMLデータベース製品カタログ 2003
## ～ネイティブXMLデータベース編～

山田祥寛

2003/9/26

## ネイティブXMLデータベース概要

**紹介する製品**

- NeoCore XMS
- Tamino
- Sonic XIS
- EsTerra XSS
- Apache Xindice

XMLの導入がビジネス局面で本格化している。XMLデータベースはXML活用において、重要な推進力であることは間違いない。

本連載では、XMLに対応したデータベース製品を、大きく「ネイティブXMLデータベース」「XML対応リレーショナルデータベース」に分類し、機能的な特性を中心に紹介することにする。

第1回の今回は、XMLストレージとして最も純粋で、さまざまなシステムへの適用も進んでいる「ネイティブXMLデータベース編」をお届けする。第2回では「XML対応リレーショナルデータベース編」を紹介していく予定なのでご期待いただきたい。

### ■なぜ、ネイティブXMLデータベースなのか

端的にいうならば、「ネイティブXMLデータベース（NXDB）」とは、Well-FormedなXMLをネイティブな――XMLツリー構造をそのままの形式で格納可能なデータベースのことをいう。データベースというと、いわゆるOracleやSQL Server、DB2に代表されるようなリレーショナルデータベース（RDB）とイコールであると思われがちだが、「データベース」という言葉は、ただ単に「データを蓄積するストア」を表しているにすぎず、RDBに限定されるわけではない。RDBは「テーブルと呼ばれる2次元表ですべてのデータが格納されるた

め、直感的で分かりやすい」「フィールド間にリレーションを設定することで複雑なデータ構造も表現できる」という利点で、爆発的に普及した。そう、一部の人間には、「データベース＝リレーショナルデータベース」という誤解を与えるほどに。

しかし、それならば、なぜいまネイティブXMLデータベースなのだろうか。本稿では、主要なNXDB製品の紹介に入るに先立って、まずはRDBでは「いけない」理由、そして、NXDBで「なければならない」理由について、簡単に紹介しておきたい。

### ■10%の不確定要素が開発の遅延を生む

「RDBではダメである」といっても、本稿では決してRDBの時代が終わったというつもりはないし――ましてや、NXDBによってRDBが置き換えられるなどと主張するつもりも毛頭ない。ただ、RDBでは明らかに対応できない局面、あるいは、RDBで「無理に」対応していた局面があったのは事実である。

ご存じのとおり、RDBは厳密なスキーマ（構成）情報を前提とするデータベースモデルである。つまり、上流設計においてテーブル・フィールド設計が不確実なままでは、下流の実装工程を進めることはできない。仮に下流工程を強引に推し進めたとしても、頻発する（であろう）設計の変更には容易に対応できないのだ。

しかし、実際の開発局面では「9割方の設計は決まった。しかし、あと1割が決まらない」というケースは少なくない。

また、システムリリース後も、9割方は恒久的に使用される情報であるにせよ、残り1割はその時々のビジネス変動によって、常に変更の可能性にさらされている流動的な要素を含む。ユーザーからの日々の改定要求によって発生するデータベーススキーマの変更、そしてそれに伴うアプリケーション対応にほんろうされている保守担当者は決して少なくないだろう。**「われわれはいつまでデータベースのお守りをしていればいいのだろう？」**。それは、常日ごろに聞こえてくる開発者の本音であるかもしれない。

そこでNXDBの登場なのである。NXDBは、必ずしも構造が明確でない半構造のXMLデータをそのまま格納できる。スキーマにとらわれないから、アプリケーション開発後（途中）の構造変更にも比較的容易に対応することができる。商品売上マスタにおいて顧客の年齢層・性別を追加したいと思ったら、XMLツリーに部分木をただ「接ぎ木」すればよい。

同様の理由で、異なるスキーマ・フォーマットを持ったデータを連携させたい場合にも、スキーマの束縛から解き放たれたNXDBでは、とにかくデータを1つのデータベースに収め、共通部分のみを縦ぐしに検索することが現実のものとなる。

以上から分かるように、ネイティブXMLデータベースは

- **構造変動が激しい**
- **管理項目の拡張が頻繁に行われる**

ような局面で有効であるといえよう。

### ■不可知のナレッジを可知（価値）あるナレッジへ

2つ目に、NXDBではこれまでリレーショナルデータベースには格納することが「できなかった」データを扱うことができる。

すなわち、帳票やドキュメント、原稿記事のような構造が極めて緩やかなデータ群がそれである。もちろん、RDBでも単純なLONGTEXTとして格納すれば、管理できないわけではない。しかし、LONGTEXTとなったデータに何の意味があるだろう。単純なプレーンテキストとして出力するのが関の山であるし、無用にデータベースを重くする要因にしかならない。そのため、このような半構造データについては、ファイルサーバやグループウェア上などで別管理し、RDBで管理するにしても、インデックス情報や属性情報などの限定された情報を「別物として」格納するのが慣例的であった。

しかし、NXDBでは上述したようなスキーマレス（あるいは、スキーマ・インディペンデント）な性質ゆえに、このような半構造データを格納するのも容易である。可視的なデータとして管理すべき部分にはタグ付けがなされているので、属性情報をわざわざ別に管理する必要はない。もちろん、管理すべき項目が新たに増えた場合にもドキュメント自体にタグを追加してやりさえすればよいのだ。

こうした「管理データのブレイクダウン」をRDBで行うには、該当のデータを削除し、あらためて分割されたデータを、新たなスキーマとともにインポートする必要があった。1度や2度ならばともかく、このようなことを日常的にやらねばならないとしたら「勘弁してもらいたい」といったところだろう。

## ■XMLドキュメントをネイティブに

ネイティブXMLデータベースという名前のそのままと思われるかもしれない。しかし、これは最も重要なポイントでもある。

昨今、システム間連携の必要が叫ばれている。企業間連携（BtoB）、企業内アプリケーション統合（EAI）をはじめ、ERPやSCMのようなキーワードもすべて根は一緒で、これまでバラバラであったシステムを統合・連携することで、業務フローの円滑化とシステム運用コスト削減を目指すものである。

このような連携局面において、もちろん、一からシステムを再構築し直し、1つのデータベースで業務情報を管理できるならばよい。しかし、それでは再構築コストがいくらあっても足りないし、スケーラビリティの制約もある。そもそも企業間連携の世界では、1つのシステムで解決しようという前提自体が非現実的なものだろう。

そのようなときにデータ交換用のフォーマットとして利用される機会が多くなってきたのがXMLである。XMLはその自己記述性と中性的な性質（環境非依存性）によって、非常な注目を浴びている。いまさら特筆するまでもなく、RosettaNetのPIPSやebXML、AribaのcXML、XML EDIなど、いまやビジネスプロトコルのあらゆる局面でXMLが導入されつつある。

しかし、このようにデータフローの媒体としてXMLが主流になっても、依然としてバックエンドシステムがレガシーなRDBであったとしたらどうだろう。XMLデータによって実現されたせっかくの柔軟性も、RDBの厳密な構造スキーマの制約を受けざるを得ない。しかも、XMLのツリーデータをRDBの2次元表にマッピングする手間は、アプリケーション層の開発に多大な負荷を課すものでもある。

しかし、NXDBを利用することで、システム間（外）でやりとりされるXMLデータを「そのまま」データベース管理することができる。このことは、ますますデータ統合・連携が当たり前となってくる今後のトレンドにあって、パフォーマンスの観点からもアプリケーション維持コストの側面からも、決して看過できない特性なのである。

## ■ネイティブXMLデータベースの主要製品

さて、NXDBの主要な特性を概観できたので、いよいよ具体的な製品紹介に移ろう。

本稿では、オープンソースを含む5つの製品を紹介する。まずは5製品の特性についてマトリックス表にまとめてみたので、ご覧いただきたい。（筆者も驚いたのだが）最初はどれも似たようなものだろうと思っていた製品が、こうして横並びに比較してみると、意外なほどそれぞれに独自色を醸し出しているのが面白い。

| 製品名 | 特徴 | リンク |
|---|---|---|
| **NeoCore XMS** | フルオートインデックスで“超”高速を実現、XRAD開発を可能とする | |
| **Tamino** | メインフレームのノウハウを受け継いだ堅牢なデータベースサーバ | |
| **Sonic XIS** | 統合的な製品ラインアップが魅力的なオールラウンドソリューション | 機能比較表へ→ |
| **EsTerra XSS** | 日本発、日本語環境に最適化されたデータベースサーバ | |
| **Apache Xindice** | Apache Software Foundationから提供されるフリーのネイティブXMLサーバ | |

**ネイティブXMLデータベース製品一覧（クリックすると機能比較表が開きます）**

では、いよいよ次ページから各ネイティブXMLデータベース製品について紹介していく。


# XMLデータベース製品カタログ 2003
## ～ネイティブXMLデータベース編～

山田祥寛

2003/9/26

## NeoCore XMS

**紹介する製品**

- NeoCore XMS
- Tamino
- Sonic XIS
- EsTerra XSS
- Apache Xindice

### ■フルオートインデックスで“超”高速を実現、XRAD開発を可能とする

「NeoCore XML Management System（XMS）」は、米NeoCore社が開発を行い、三井物産／三井情報開発が販売代理店となるネイティブXMLデータベースサーバである。国内実績という意味では、2003年4月に国内販売を開始してまだ日は浅いものの、すでに導入ベンダ数15社を数える。急速に普及しつつある製品である。

NeoCore社は、もともとが数学理論を専門とするラボを前身とする歴史的経緯もあり、数学的な演算を前提に置いた検索技術では一日の長がある。NeoCore XMSの特徴とは、この検索技術を最大限に引き出した驚異的なパフォーマンスとスケーラビリティの実現にあるといえよう。

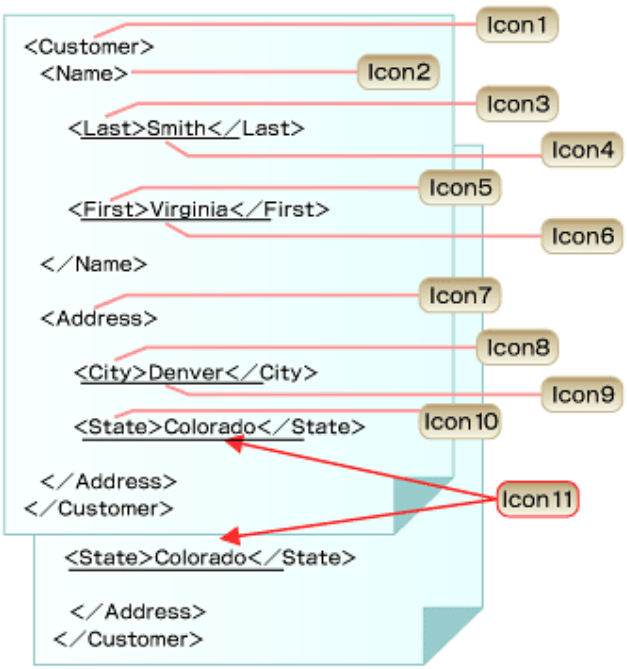
### ■最大3万倍の高速アクセスを実現するインデックス技術「DPP」

NeoCore XMSは、データ保管時にすべての要素に対してインデックス付けを行う「フルインデックス」方式を採用している。つまり、開発者はデータベースチューニングにおいて、これまで常に悩まされてきたインデックスキーの検討から解放されるというわけだ。NeoCore XMSでは、このインデックス方式はDPP（Digital Pattern Processing）と呼ばれている。

以下にNeoCore XMSのパフォーマンスを裏付けする検索メカニズムについて、その概略を紹介することにしよう。

**「ロケーションパス＋データ」を64bitの固定長で表現**

XMLドキュメント中のノードは、ノードに至るまでの「ロケーションパス（階層構造）」とデータ本体で表現することができる。NeoCore XMSでは、この「ロケーションパス＋データ」を独自のアルゴリズムを用いて、64bitの固定長データに変換する（変換された固定長データを「アイコン」と呼んでいる）。




**XMLのエレメントごとにアイコンが生成される**

DPP技術ではどのような長さのデータであっても、必ず64bitのアイコンに変換することができる。64bitというと、いかにもすぐに重複が発生するように思われるかもしれないが、ベクトル理論を利用して表現されたアイコンでは、その可能性は限りなく小さい。異なるデータが同一のアイコンに変換される可能性は、実に$18\times10^{18}$分の1なのだ。

生成されたアイコンはそのままインデックスとして、NeoCore XMS上に自動登録される。

**パターン検索に要する時間は、1.5メモリ・サイクル**

XPath／XQueryによる検索に際しては、指定された条件式が、インデックス作成時とまったく同じアルゴリズムを用いて、アイコンに変換される。つまり、NeoCore XMS内部ではアイコン同士のパターンマッチングを数学的に、かつダイレクトに行うことができるというわけだ。所要時間は平均で1.5メモリ・サイクルであり、しかも、データ量に依存せず一定のパフォーマンスを発揮することができる（理論的には、4E（エクサ）bytes（$10^{18}$）まで一定のパフォーマンスを実現可能）。インデックス自体はフラットな構造であるので、XMLドキュメント自体のデータ階層にも依存しない。

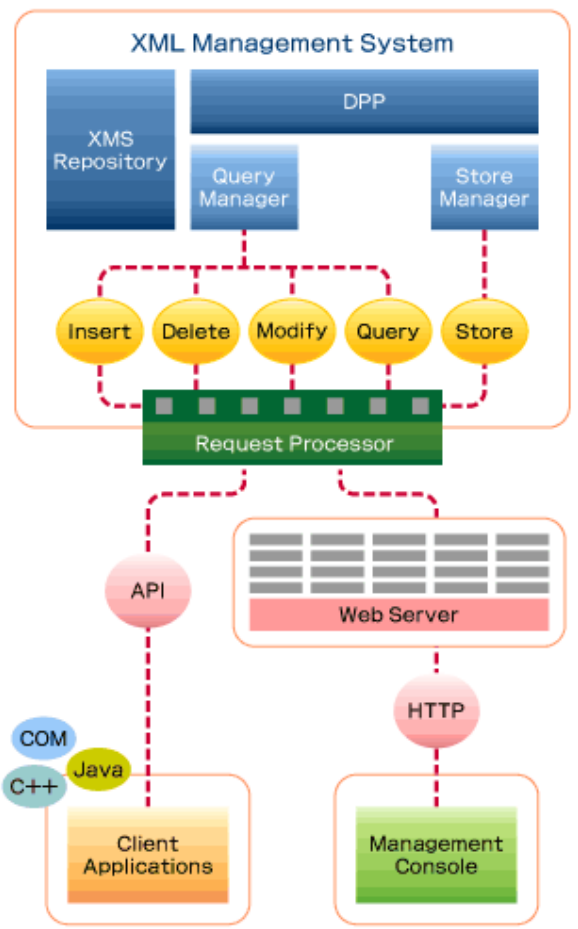
もちろん、検索条件によっては必ずしも完全一致するデータばかりではない。部分一致検索の場合などには、DPP技術による絞り込みがあらかじめ行われたうえで、通常の検索技術が併用される形となる。

**更新処理にも有効なインデックス**

インデックスというと、とかく検索処理のためのデータテーブルであると思われがちだが、NeoCore XMSにおいては更新・削除時の（挿入・削除）データ位置特定に際しても、インデックスが利用されているようだ。上述のとおり、インデックスによるパターンマッチングはほとんど無視できる時間であるので、データ容量の増加に際しても、レスポンスタイムを比較的緩やかな上昇カーブに抑えることができる。

## ■NeoCore XMSを支えるシステム構成

NeoCore XMSは、ほかのネイティブXMLデータベースと比較しても、極めてコアなモジュールからなるコンパクトな構成が特徴といえる。以下にNeoCore XMSを構成する主要な構成を挙げてみた。




**NeoCore XMSのシステムイメージ図**

### XMS Repository／DPPエンジン

NeoCore XMSのコアエンジンというべきデータストアと処理エンジンである。前項で詳細はすでに述べているので、割愛する。

### Query Manager／Store Manager

データベースの作成・削除からバックアップ処理、データ本体の更新・削除などを行うモジュール。NeoCore XMSでは、標準でコマンドラインベースのNeoXML Utilsが用意されており、これを利用することで一元的にデータベースを管理できる。

また、プラグインを導入することで、XML開発環境「XMLSpy」（開発元：Altova、国内販売元：東芝ITソリューション）からNeoCore XMSにアクセスすることも可能である。

### XMSコンソール

HTTPプロトコルを用いたアクセスを管理する。NeoCore XMSではHTTPプロトコルによる通信手段を備えており、WebブラウザやAPI経由のクライアントアプリケーション、先述のXMLSpyなどからアクセスすることができる。XMSコンソールはWebベースの管理ツールで、これを用いることでWebブラウザからデータベースの管理やXMLドキュメントの基本的なオペレーションが可能となる。

以上からも分かるように、NeoCore XMSの構成は極めてシンプルである。NeoCore XMS自体は、ほかの製品に見られるような統合開発環境やリレーショナルデータベースとのマッピング機能、日本語全文検索機能などを提供しない。これら付随的な機能については、サードベンダから提供されているので、すでにメジャーとなっている製品と連携すればよいというのがNeoCore XMSの基本ポリシーだ。同様の理由で、アプリケーションフレームワークも提供していない。アクセスのためのAPIが用意されているのみである。

また、NeoCore XMSはXML SchemaやDTD、RELAX NGなどによるデータ検証にも対応していない（つまり、アプリケーション層で実装する必要がある）。自己構成型データベースであるからスキーマ情報は不要というのがNeoCore XMSの考え方なのである。

これは機能不足ではなく、NeoCore XMSがユーザーに与えた「自由な選択肢」であると筆者は考える。各ベンダからXML関連製品、アプリケーション開発環境が多くリリースされている現在、後発のNeoCore XMSがあえて類似した製品群を投入する価値はあまりないだろう。むしろXMLストレージとしてのコアな機能を軽量な形で提供することで、常にハイパフォーマンスを維持する――NeoCore XMSの姿勢はその点で極めて一貫している。これは現場のユーザーにとって最も必要なソリューションであるかもしれない。

## ■NeoCore XMSの超高速が可能にするXRAD開発

NeoCore XMSがDPP技術と並んで大々的にアピールしているのが、システム開発の新手法である「XRAD（eXtreme Rapid Application Development）」開発だ。XRADとは、文字どおり「“超”高速のアプリケーション開発」のための手法である。自己構成型データベースの長所を十分に活用することで、データベース開発の負荷を大幅に軽減するというものだ。

従来のシステム開発においては、データベース（スキーマ）設計ができていないがために、開発フェイズに移行できないケースが少なくなかった。また、スキーマ設計の変更が、開発途中（または開発済み）のアプリケーションに対して与えるインパクトは極めて大きい。いわゆるデータベース設計・開発にかかわる作業は、システム開発全体の30～70%を占めるといわれるから、開発コストを左右する最大の要素といっても過言ではないだろう。

しかし、自己構成型データベースであるNeoCore XMSを利用すれば、事前のスキーマ定義は不要だ。DPP技術の超高速性を活かせば、データ量の拡張に際しても、インデックス・チューニングなどは必要ない。また、データベース上で各要素間の関連性、データマイニングが高速かつ簡単に行えるので、データを分析しつつ、イテレーティヴ（反復的）な開発も可能となる（それは複数のデータベースから抽出した異なるフォーマットのデータについても同様だ）。つまり、XRAD開発を利用すれば、パイロット的にスタートしたシステムをベースとして、実運用システムへと移行するという流れが現実のものとなるのだ（スパイラルアップ）。

もっとも、こうして見ていくと、XRAD開発は、イコールDPP技術の“超”高速性をただいい換えているだけに見えるかもしれない。実際、それはしかりである。しかし、開発の現場に求められているのは複雑な小難しい概念ではない。このシンプルな概念がこれまでの開発パラダイムを大幅にシフトした点にこそ、NeoCore XMSの真価があるといえよう。