December 2003

A Scalable Approach to Processing Large XML Data Volumes

Tim Weitzel
University of Frankfurt

Thomas Tesch
Infonyte GmbH

Peter Fankhauser
Fraunhofer IPSI

A SCALABLE APPROACH TO PROCESSING LARGE XML DATA VOLUMES

Tim Weitzel
Institute for Information Systems
University of Frankfurt
tweitzel@wiwi.uni-frankfurt.de

Thomas Tesch
Infonyte GmbH
tesch@infonyte.com

Peter Fankhauser
Fraunhofer IPSI
fankhaus@ipsi.fhg.de

Abstract

The emerging penetration of IT architectures with XML leads to increasing XML data volumes. Available tools often fail in realizing scalable XML processing for large XML data volumes. This paper introduces Infonyte-DB, a persistent XML Processor that economizes on system resources and allows processing large XML data volumes. Based on concrete application scenarios it is illustrated how Infonyte-DB can be deployed for XML based web services, technical documentation, and mobile information management.

Keywords: Scalable XML processing, DOM

Introduction

The Extensible Markup Language (XML) is increasingly used in its original core domain content management as well as in other contexts ranging from B2B message exchange (Rawolle et al. 2002) to converting heterogeneous data sets (Buxmann et al. 2001). More generally, XML is used as platform and device independent base technology for data exchange and in Web Service architectures (Beimborn et al. 2002). In this paper, the role of XML for loosely coupling IT systems and limitations of existing XML tools when processing large data volumes are discussed. Since most available XML processors simply cannot cope with large XML file sizes, a virulent question for the use of XML in corporate IT infrastructures is how to efficiently store and process XML data. A recent example includes straight through processing (STP) initiatives for and XML-based end-to-end integration in cross national securities business. Here, large volumes of FIXML and SWIFT data (or swiftML, respectively) need to be processed in real time. As will be discussed later, 1 MB of SWIFT data is about 10 MB of swiftML (same content); and since 1 MB of XML requires up to 20 times its size of main memory (traditional DOM implementation, XSLT transformations), this means that our 1 MB of original SWIFT data translates into 200 MB of main memory requirements for XML processing.

To address this challenge, the scalable XML processor Infonyte-DB was developed. Infonyte DB is a novel tool for the scalable processing of XML that integrates seamlessly with existing IT architectures. It provides a native XML store and supports efficient and scalable XML processing with the prevalent XML transformation and query languages recommended by the World Wide Web Consortium (W3C). Infonyte DB processes any well-formed XML without being constrained by a particular schema or DTD. Thereby, typical weaknesses of traditional database technology for XML processing, including the need for rigid schema definitions and complicated mappings to a different data model, are avoided.

The architecture of Infonyte DB is highly modular and integrates seamlessly with existing environments. Based on the Infonyte PDOM, a persistent implementation of the DOM (Document Object Model) that was developed at Germany's main think tank Fraunhofer Institut für Integrierte Publikations- und Informationssysteme (IPSI), additional modules (XPath, PXSLT, PDOM Collections, and XML Workbench) can be flexibly assembled according to customer demand. The backend-interface to the data server is open to allow integrating existing physical storage layers. Implemented in 100% Java, Infonyte DB is fully platform

independent. The entire functionality can be utilized via Java APIs. In addition, Infonyte includes Servlet- and JSP-templates as a basis for realizing web-enabled XML applications. Historically, the Persistent DOM and its succeeding Infonyte XML processor result from a decade of research on OO databases at Fraunhofer Institute. Infonyte technology is used in a wide variety of applications and products, including web portals, content management systems and mobile information systems. Especially customers in the American aviation industry and recently from the IT and finance sector use Infonyte-DB as core XML component for their products.

Challenges for XML Processing

The cost efficient and flexible integration of heterogeneous systems is a key challenge for IT systems architectures. Particularly, media and process discontinuities need to be overcome. In this context, XML prevailed as a neutral data exchange format. Using XML, even complex data structures and meta data can be described and exchanged between different platforms, applications and programming languages. Also, for practical applications, there is a cornucopia of commercially and freely available tools for authoring, validating and transforming XML. The benefits associated with XML are most evident in two application domains: *media independent publishing* and *coupling of business processes.*

Media independent publishing: Due to its inherent separation between content and layout, a core application domain of XML are complex cross-media publishing processes. XML is a perfect core technology for content management systems aggregating different content for particular media as well as for document management systems focusing on managing, archiving and retrieving large document collections. For these tasks, proprietary data formats have to be transformed into an XML representation and then further processed. Using transformation languages like Extensible Stylesheet Language Transformations (XSLT), from these XML documents any presentation format (HTML, PDF, eBook formats) can be generated.

Coupling of business processes: Within enterprises, the coupling of business processes to overcome heterogeneities between mainframe systems, legacy applications, ERP software and web applications is discussed as Enterprise Application Integration (EAI) (Linthicum 2000). Between enterprises, i.e. the cross-firm process integration (B2Bi) is largely aimed at integrating different partners, data formats, technologies and processes. Traditional integration concepts are based on converting the proprietary data formats of all sub processes into each other or – using traditional data base systems – on developing a unified data format for the entire process or firm. Both alternatives require substantial developing time and lead to complex and convoluted architectures. Due to the combination of data and associated metadata, XML allows for incremental and demand-oriented integration approaches as a best of both worlds approach: data from all different sources can be dumped into an XML data warehouse that is completely decoupled of business processes. In that warehouse, the XML data can then be validated, filtered, compared and – quite comparable to media independent publishing – transformed according to the requirements of the respective target application systems using well established XML standards like XSLT, XPath and XML Schema. Using this approach mutual dependencies can be minimized and systems can be loosely coupled (Hasselbring 2000).

But the increasing penetration of XML based integration architectures as described above goes along with an increase in XML data to be processed. Unfortunately, available XML tools cannot cope with these data volumes, mainly for two reasons:

- *Verbosity:* The main advantage of XML, i.e. combining data and metadata, is at the same time accountable for the resulting large document sizes. Accordingly, it is no exception to find XML documents with more meta data than original content data. Hence, directly processing textual XML is a heavy burden for both, storage systems and bandwidth.

- *Scalability:* Available XML tools suffer from high memory requirements and low processing speed. Essentially, this results from the way DOM (Document Object Model) interfaces are implemented, creating a main memory DOM tree representation of the entire document. Depending on the particular XML document and DOM implementation, the original textual XML document can consume twenty time its size in a main memory DOM. Analogous problems are found with XSLT implementations.

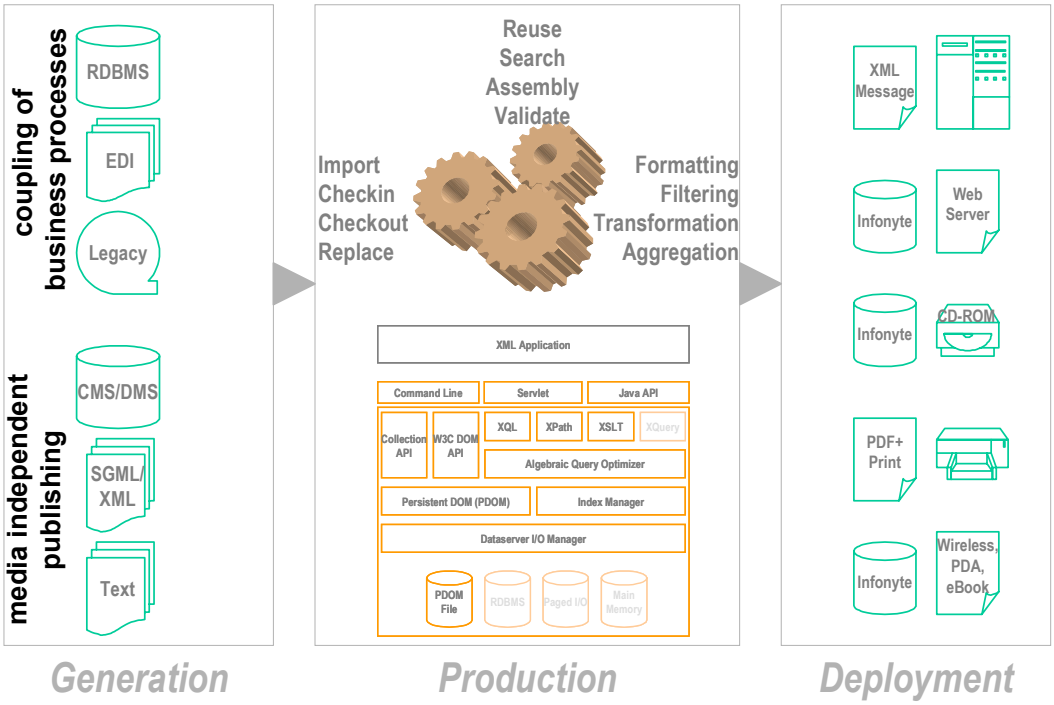

Figure 1. XML Processing with the Infonyte Processor

One way of addressing the aforementioned challenges, i.e. XML mass data processing, are native XML database systems or relational database systems with XML extensions. Both approaches focus on storing large numbers of XML documents while offering various search possibilities. When using XML for integration purposes, though, it is not storing but rather scalable processing that is key. Besides authoring tools, publishing systems as one of the earliest XML strongholds require efficient data filtering, transformation and formatting for generating target formats. The same accounts for data converting. It is not data storage but efficient matching with data sources and the subsequent aggregation and transformation that is important. These requirements ask too much of pure XML storage systems and traditional DOM and XSLT implementations. Figure 1 illustrates the role of IDB for XML processing in the context of media neutral publishing and business process integration.

In the next section, we describe the persistent XML processor IDB that is particularly powerful for media independent publishing and integration processes. IDB focuses on XML processing. Accordingly, storing XML data is merely seen as a precondition to enable scalable XML processing in a gigabyte range even on standard PC hardware. Even existing XML applications and tools using DOM or XSLT standard interfaces can become scalable XML applications but exchanging the respective modules with the Infonyte solution.

Infonyte-DB

IDB was designed to provide a modular and highly scalable XML kernel that can easily be extended according to the requirements of the applications using it. To ensure easy integration with other applications and processes, all communication between IDB and eternal applications uses standard interfaces. From a user's perspective this guarantees a risk free investment even in highly innovative technologies since no costly investment in proprietary interface adaptation is necessary.

Overview

In figure 2, the IDB architecture is visualized. Due to its modularity, most components can be used independent of one another. The core element is a persistent DOM (PDOM). Based on PDOM, XML documents and document collections in the multiple

Gigabyte range can be created, explored, queried, and modified with constant and moderate main memory consumption, utilizing the rich family of XML standards. PDOM is a persistent implementation of the W3C DOM API for XML that implements standard query and transformation processors. While using PDOM characteristics for optimization purposes, these can also be used with any other DOM implementation. In contrast to other DOM implementations, Infonyte PDOM provides a fully compliant DOM 2 API and addresses typical shortcomings associated with traditional DOM implementations by ist compact binary format, efficient node access, query and navigation. Especially for processing large XML documents or document collections (500+ MB), indices can be defined and used with the collection manager depicted in figure 2.

IDB is implemented in 100% Java and runs on basically all relevant platforms without any problems. Using Java turned out to be an advantageous choice due to Java's dominant position for internet-based server applications and since performance is no issue any more. Also, in contrast to static compilers, existing just in time compilers can use additional run time information for optimization. IDB components are lightweight, with code size between 400 and 800 KB and minimum memory requirements of no more than 16 MB.

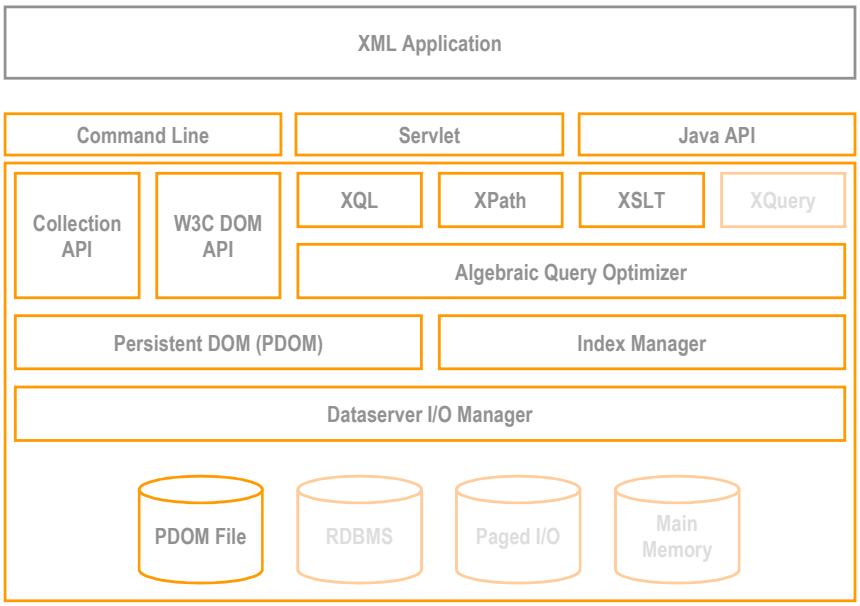

Figure 2. Infonyte Component Architecture

The entire functionality can be utilized via Java APIs. In addition, Infonyte includes Servlet- and JSP-templates as a basis for realizing web-enabled XML applications. IDB can be called directly by applications using the command line, a web server or Java interfaces, and it directly processes well-formed XML without the necessity of declaring a schema or DTD. All index and storage structures are generated using document instance information thereby avoiding the costly need for complex mappings to physical data models known from storing XML content in relational systems and also some XML data bases.

In the next section, the core components of IDB are explained in more detail.

Persistent DOM (PDOM)

The core of IDB is Infonyte PDOM, a persistent implementation of the W3C DOM API for XML. With Infonyte PDOM you can create, explore, query, and modify XML documents in the multiple Gigabyte range with constant and moderate main memory consumption.

DOM is the World Wide Web Consortium's (W3C) standard programming interface for XML documents. Using DOM, an XML document is represented as a tree with different node types modeling the particularities of elements, attributes etc. There is a wide variety of DOM implementations for all relevant programming languages. They usually generate the DOM tree in main memory

resulting in substantial memory problems. Using main memory DOM implementations, a 20 MB XML document requires about 200 and 400 MB of main memory.

Infonyte-PDOM uses a different approach: While providing a fully compliant DOM 2 API, XML documents are stored using a compact, lossless binary format the storage layout of which is optimized according to XML's tree structure. This format contains indices for document structure and sequence for efficiently evaluating XPath expressions and for addressing document nodes using simple integer arithmetic. By factorizing redundant information, even when considering the additional indices, overall memory consumption is between 30% and 100% of the original XML document size. An early realization of this format is described in (Huck et al. 1999). The compact binary format and memory layout optimized for XML enables extreme IO performance. An LRU cache uses physical memory segments with a constant number of DOM nodes. Also, synchronization mechanisms for multithreading access and maintenance and defragmentation functions for the persistent documents are supported.

All in all, the Infonyte-PDOM is lightweight, fast and reliable:

- It is lightweight: DOMs are stored using a compact, lossless binary format that can reduce the size of serialized XML down to 30%. Together with its self-optimizing in-memory representation, its LRU cache, and its consequent lazy evaluation strategy, the Infonyte PDOM enables fully-fledged DOM applications on very large documents with main-memory requirements down to 16 MB.

- It is fast: Structural indices and configurable full text and data indices support efficient node access, navigation, and evaluation of XPath expressions, without impeding performant bulk loads and nodewise updates. Multi-user queries scale due to its full support of Java threads.

- It is reliable: Infonyte PDOM's recovery manager supports 2-phase-commit to guarantee atomic and durable updates on multiple documents. Building on Java synchronization Infonyte PDOM also supports isolated multi-user updates.

Data Server

The data server module implements the actual IO on the storage medium and provides access to data segments of any size. The data server interface is completely open and easily allows individual implementations. The data server implementation distributed with Infonyte-DB manages memory segments using binary files optimized for XML and efficient IO. As can be seen in the architecture in figure 2, data server implementations can also be used on relational databases or for realizing main memory data bases (Manegold 2000). Besides, a data server could access its data by web services to enhance scalability in a multi user mode. Since the openness of the backend interface allows for developing hardware specific data servers, IDB can easily be adapted for use with systems as diverse as handheld and NAT server.

XML Queries and XSLT Transformation Using PXSLT

IDB supports all major XML query languages and transformation tools. Singular or multiple documents can be queried, search results aggregated and logical views realized. Extensible Query Language (XQL) is a predecessor from 1999 that never received W3C recommendation status (Robie et al. 1999). At present, the dominant XPath is used within XSLT for selecting document fragments. Also, the future XQuery standard is expected to play an important role (Clark and De Rose 1999) (Draper et al. 2002). Both, XPath and XQL statements are translated in an initial execution plan by IDB which is then represented using an XPath like algebra. This execution plan is optimized by applying transformation rules for minimizing execution time (logical optimization). The subsequent physical optimization using the index structures described above results in an optimized DOM operations sequence to instantiate the query results. For the first time, the Infonyte XSLT processor („Infonyte-PXSLT" for Persistent XSLT) allows to directly process XSLT on a persistent representation of XML documents. Especially when using XSLT on large XML documents there have been massive problems when using main memory based implementations. As a consequence, in practical applications XML documents are often cut into pieces of operable size using a sequence ox XSLT scripts. In contrast, the PXSLT processor can easily process XML document in the multiple Gigabyte range at constant main memory consumption. Using the compact PDOM format with its optimized caching procedures this can be done without processing time losses. The imminent XQuery standard is already depicted in the architecture in figure 2. XQuery uses the XPath standard for selecting document fragments and extends it by SQL like constructs for combining and restructuring document fragments. Query results are represented using XML. In addition, XQuery defines necessary functions and operators all XML schema data types. For a prototype based on the DOM API see Fankhauser et al. (2002).

User-Defined Indices

Structural indices are not sufficient for content-based queries to large XML documents and document collections. In order to also capture text node values in an index, user-defined indices can be defined. Depending on the value type to be indexed, integer, double, or string indices are generated. For text-based search, generating word indices is possible, too. Index entries can either point to indexed nodes or other elements, e.g. the root element. In doing so, word indices even for very large document sets can be defined directly returning documents. In addition, users can extend existing index structures by types like currency format, for example.

Performance

Experiments using an XML version of the freely available freeDB CD database (FreeDB 2002) give an impression about IDB performance. FreeDB consists of about 500,000 CD descriptions. The XML version is about 500 MB. The XML version as PDOM can be downloaded at **www.infonyte.com** ➔ download.

On a standard PC (1,8 Ghz, 512 MB RAM) parsing and PDOM creation (32 million XML nodes, 400 MB) including all structural indices takes about 4 minutes (~2MB/s). Generating the user-defined index for all CD keys indexes 548,000 nodes or 1.7% of the entire database in about 88 seconds. Generating the full-text index (28 million nodes, 89% of the entire database) takes 17 minutes, resulting in an index size of 90 MB. Depending on the respective index definition, between 5 MB and 10 MB raw XML text is indexed every second. XSLT processing, for example for generating HTL, throughput is up to 10 MB per second. When searching for CDs with particular titles or tracks using the full-text index, first results are delivered within 5-10 milliseconds, analogous for subsequent hits. Processing AND-queries is more complex since intersections of partial results need to be generated. Figure 3 depicts search results for “bowie” on “bbc” in XML in the freeDB-demo available at the Infonyte website.

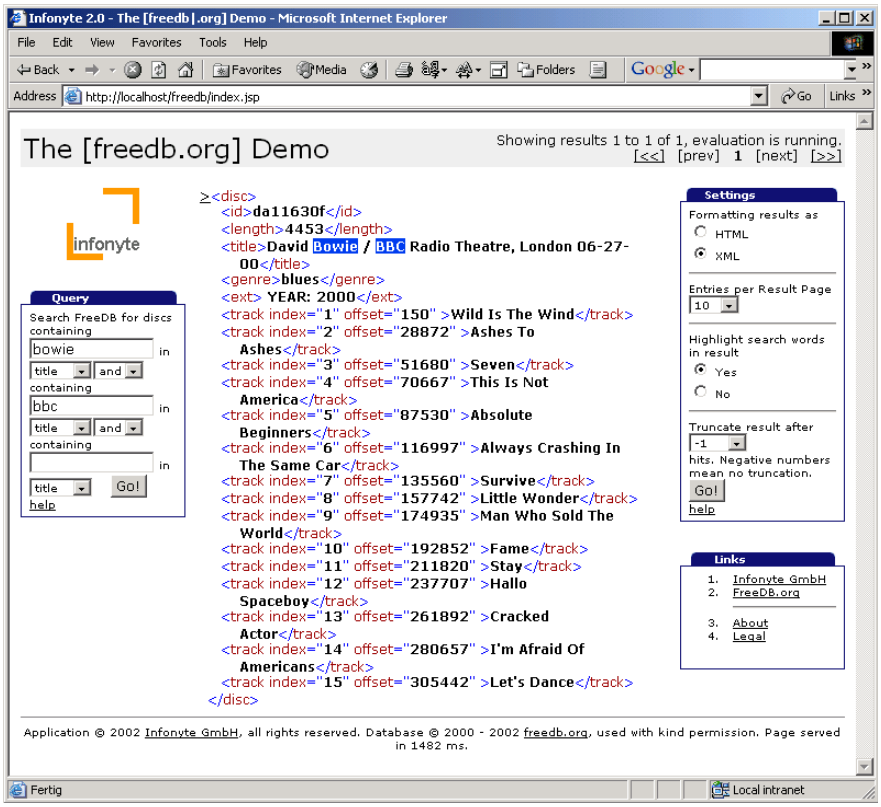

Figure 3. Search Results (XML) in FreeDB Demo

A look at main memory and CPU usage demonstrates the scalability of the concept (figure 4). It shows that parsing and PDOM creation (left) goes along with a constantly low main memory load while fully utilizing the CPU. In contrast, a traditional main memory DOM implementation (right) first bombs main memory before the process is terminated before completion.

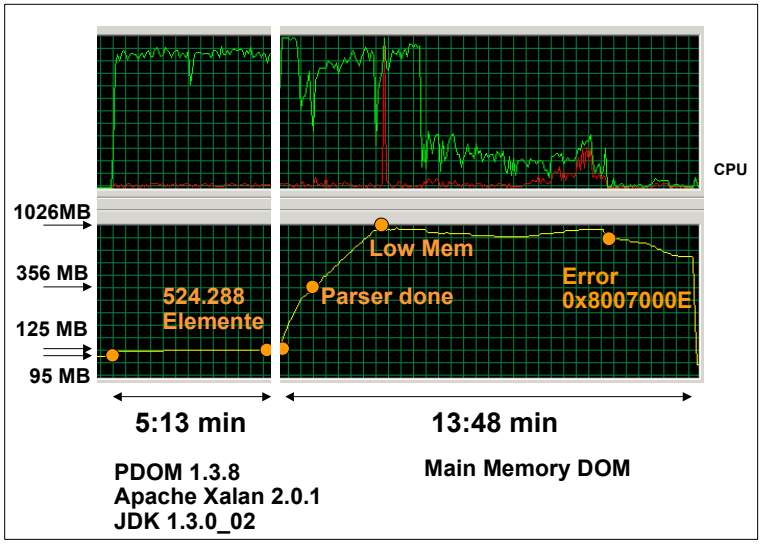

Figure 4. Main Memory Consumption of PDOM versus Main Memory DOM

Application Scenarios

In this section, we describe three IDB application scenarios. It becomes clear that for coupling business processes and for media independent publishing for interactive electronic technical documentations, and for mobile information management the IDB architecture offers genuine advantages that directly translate into business value.

XML Warehouse

A typical XML application scenario for business process integration is an XML warehouse for congregating data from different information systems into one common XML representation. All data is then reformatted, for example for publishing on a web server, using XSLT or XQL/XPath commands.

In our case, a huge US-based financial information and service provider needed to organizationally as well as technically separate the XML warehouse from the business applications as fully as possible, among others to guarantee separate operations. That is why solutions providing merely logical XML views are inappropriate. Based on IDB, an application was developed for individualized messaging and feeding a web portal that allows customers to get their individual transaction data in real time. The Infonyte system has to get 10 GB XML raw data every day, index it and make it available for ten days. Easily, IDB could be integrated as scalable XML backend into the J2EE conforming IBM WebSphere Application Server. Especially the capability to straightforwardly process these large amounts of data going along with access time in a millisecond range for dynamically generating individualized Web pages made IDB an ideal technological solution for this problem domain. Here, automating business processes formerly focused on generating and printing individual reports (mainframe application) on inexpensive PC hardware allowed significant cost improvements.

Interactive Electronic Technical Documentation (IETD)

In the aviation industry, there is a long tradition of SGML based technical documentations. Due to cost considerations, among others, many systems are redesigned as browser based XML applications. The main challenges are designing a distributed authoring environment with a centralized data repository and an efficient production process for compiling and formatting

electronic manuals for different user groups, providing powerful reading and navigating tools, and enabling an efficient and secure distribution. Due to its easy integration capability and the possibility to efficiently work with very large documents, Sikorsky Aircraft Corporation developed an XML-IETD system based on Infonyte. IDB is used for the production process as well as for providing the documents via a web server. For the production task, especially the Infonyte XSLT processor is the key element for the demand driven compilation of large XML data volumes. For the subsequent usage of the technical manuals in a reading environment, Infonyte is used as client-side tools to enable XML query languages to retrieve relevant document fragments.

This architectures helped Sikorsky realize substantial cost and service improvements. Among others, document maintenance becomes much simpler since service personnel can include change proposals or mainztenance reports directly into the system using the PDOM update capability. The airline sector is predisposed for cost efficient IETD system on XML basis because of the historic predominance of SGML. One the one side, in many cases client-side standard software (e.g. XML/PDF browsers) can be used, making elaborate software development obsolete. On the other side, the overall development process is faster and less expensive using XML tools. Similar application scenarios can be found in the area of form-based document processing and editing other technical literature and reference books, as for example medical or legal reports.

Mobile Information Management

Low memory consumption, platform independence qua Java and the compact PDOM format make Infonyte the ideal XML based mobile application kernel. US-based Vaultus (**http://www.vaultus.com**) has used Infonyte technology as foundation of their mobile information platform. In addition to data management, the system offers offline capabilities, secure transactions, network independence, and remote maintenance services.

Figure 5. Mobile Sales Force Automation Based on Infonyte

The reason for employing the Infonyte XML processor was mainly its economical usage of system resources on mobile devices. The application directly manipulates the data in XML, comparable to using relational data base systems. Therefore, updates on a highly granular document structure as provided by Infonyte PDOM are a key capability in this application domain. In the future, the compact PDOM format could be used for directly exchanging XML data when synchronizing. This could completely avoid parsing the documents and all data could be instantly and efficiently accessed using the encoded index structures. Figure 5 shows a screenshot of a mobile sales force automation software based on the Vaultus platform.

Figure 6. FreeDB on an iPaq

To test the performance of the Infonyte architecture on a small device, we developed a mobile demo scenario using the full freeDB as described in section 4. In a limited version consisting only of the data server, the PDOM, and the index and collection APIs (all in all about 300 KB), the full FreeDB demo runs on a PocketPC (iPAQ Pocket PC H3800 with 64 MB Ram, 32 MB Rom, 206 MHz ARM-Processor, 1GB IBM-Microdrive, Personal Java 1.2 Insignia Jeode). Using the indices, response time for Boolean search on this limited platform is 1-2 seconds, searching for singular criteria is even faster. Figure 6 shows results for "bowie" on an iPaq

Conclusions

Scalable XML processors like Infonyte-DB offer innovative ways of handling large XML data volumes and can help overcome scalability problems inherent in many existing XML tools. Experiments with open source implementations (e.g. FOP processors, XML editors, WebDAV servers) illustrate that by simply exchanging the respective storage backend with the Infonyte solution can transform any traditional XML application into a highly scalable XML application. Focusing on a lean XML kernel and a highly performing implementation in a platform independent language like Java make IDB a universal tool for XML applications. The ubiquitous need for making existing application XML capable and the easy integration offered by IDB's modularity and standard conformance also make it a cost efficient add-on to many existing applications.

A free Infonyte DB for evaluation, including the FreeDB demo, is available at **http://www.infonyte.com**.

References

Beimborn, Daniel, Mintert, Stefan and Weitzel, Tim: Web Services und ebXML. In: WIRTSCHAFTSINFORMATIK (2002) 3.

Bray, T., Paoli, J. and Sperberg-McQueen, C. M.: Extensible Markup Language (XML) 1.0. W3C Recommendation. 10. Februar 1998. **http://www.w3.org/TR/1998/REC-xml-19980210.html**.

Buxmann, Peter, Ladner, Frank and Weitzel, Tim: Anwendung der Extensible Markup Language (XML): Konzeption und Implementierung einer WebEDI-Lösung, In: WIRTSCHAFTSINFORMATIK (2001) 3, pp. 257-267.

Clark, J. and De Rose, S.: XML Path Language (XPath) Version 1.0. W3C Recommendation. 16 November 1999. **http://www.w3c.org/TR/xpath**.

Clark, James: XSL Transformations (XSLT) Version 1.0. W3C Recommendation. 16. November 1999. **http://www.w3c.org/TR/xslt/**.

Draper, D., Fankhauser, P., Fernández, M, Malhotra, A., Rose, K., Rys, R., Siméon, J. and Wadler, P.: XQuery 1.0 Formal Semantics. W3C Working Draft. 26. März 2002. **http://www.w3.org/TR/query-semantics/**.

Fallside, David C.: XML Schema Part 0: Primer. W3C Recommendation, 2. Mai 2001. **http://www.w3.org/TR/xmlschema-0/**.

Fankhauser, P.; Groh, T. and Overhage, S.: XQuery by the Book: The IPSI XQuery Demonstrator. EDBT 2002: 742-744. **http://xml.darmstadt.gmd.de/xquerydemo/**

FreeDB: **http://www.freedb.org**

Hasselbring, W.: Information System Integration. In: Communications of the ACM 43 (2000) 6, pp. 32-38.

Huck, G., Macherius, I. and Fankhauser, P.: PDOM: Lightweight Persistency Support for the Document Object Model. OOPSLA Workshop "Java and Databases: Persistence Options". November 1999. Denver, Colorado, USA.

Le Hors, A., Le Hégaret, P., Wood, L., Nicol, G., Robie, J., Champion, M. and Byrne, S.: Document Object Model (DOM) Level 2 Core Specification. W3C Recommendation. 13. November 2000, **http://www.w3.org/TR/DOM-Level-2-Core/**.

Linthicum, D. S.: Enterprise Application Integration, London 2000.

Luoma, R.: Using XML to Enable Low-Cost Deployment of Content at Lockheed Martin Aeronautics. XML 2001 Conference and Exposition. December 2001. Orlando, Florida, USA.

Manegold, S.; Boncz, P.A. and Kersten, M.L: Optimizing database architecture for the new bottleneck: memory access. VLDB Journal (2000) 9(3), pp. 231-246.

Network Working Group: HTTP Extensions for Distributed Authoring – WEBDAV. RFC 2518. **http://www.ietf.org/rfc/rfc2518.txt**.

Rawolle, J.; Ade, J. and Schumann, M: XML als Integrationstechnologie bei Informationsanbietern im Internet - die Fallstudie BertelsmannSpringer, In: WIRTSCHAFTSINFORMATIK (2002) 1, pp. 19-28.

Robie, J., Lapp, J. and Schach, D.: XML Query Language: A Proposal. W3C-QL '98 workshop proposal. **http://www.w3.org/Style/XSL/Group/1998/09/XQLproposal. html**.