

SYBASE®

**MANAGING XML WITH
ADAPTIVE SERVER® ENTERPRISE**



TABLE OF CONTENTS

INTRODUCTION	3
XML DATA MANAGEMENT—AN INTEGRAL PART OF DBMS	3
XML—INFORMATION INTERCHANGE BACKBONE FOR APPLICATIONS	3
XML HANDLING IN APPLICATIONS—BOTTLENECKS	4
ROLE OF DATABASES IN MANAGING XML	5
OVERVIEW OF XML DATA MANAGEMENT SUPPORT IN ASE	9
USAGE SCENARIOS	11
AREAS OF APPLICATION	14
SUMMARY	16

MANAGING
XML

INTRODUCTION

In the world of IT, XML began as a universal format for structured documents and semi-structured data, and has now grown into a term that encompasses a family of exciting new technologies. Initially, it helped separate the presentation aspects from the actual content and provided a way for storing Web content. It quickly became the lingua franca for businesses exchanging information electronically. Its simplicity, broad support, and low cost of implementation drove the rapid adoption of XML. Today, many technologies built on XML are becoming pervasive in the industry.

The challenge to leveraging the power of XML is finding the best solution for transparent, inexpensive generation and management of business-critical XML data without writing a lot of custom code. This paper highlights the importance of managing XML data (storage, index, query, and transformation) at the database level, and discusses the capabilities of the XML Management Package for Sybase Adaptive Server Enterprise. It also illustrates a few use cases for the XML Management Package and presents various areas of application for ASE XML data management functionality.

XML DATA MANAGEMENT—AN INTEGRAL PART OF DBMS

XML—INFORMATION INTERCHANGE BACKBONE FOR APPLICATIONS

Today, when XML is used as a platform-independent data exchange format, it can assist in solving a number of problems. XML is beginning to play a role in a number of industries including the financial services industry. Most notable applications of XML include:

Real-Time Integration: One of the primary reasons that companies are adopting XML is to support their systems-integration strategies. Financial institutions are discovering that offering specific services to customers, business partners, and suppliers on the Web requires access to multiple systems in real time. XML provides the best solution for economically integrating these existing systems and enabling integrated business services to be delivered via the Web. For example, as financial institutions begin to use customer relationship management (CRM) tools to manage customer data and services, they need to tie together data and capabilities from multiple systems. These enormous data consolidation and systems integration requirements can only be fulfilled with a common XML standard. This is especially true in the financial-services industry because of the high volume of data and the large number of systems involved. XML allows the CRM systems to be readily integrated with other systems. Once the systems are integrated, CRM users can access data and make requests from multiple systems in real time.

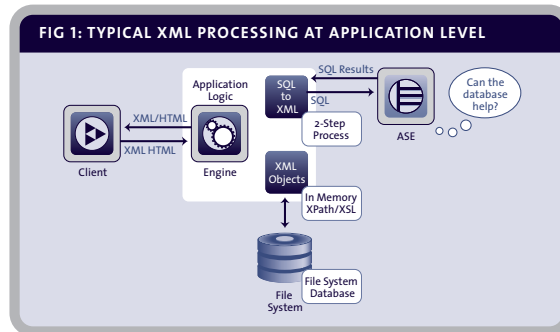
Global Straight-Through Processing (GSTP): An industry-wide initiative necessary to implement the SEC-required T+1 settlement cycle, GSTM requires multi-system and multi-enterprise integration coupled with real-time decision-making and fulfillment. Another significant problem within any straight-through processing environment is single representation

of the data and therefore a single data format. XML reduces the risk of data error and reduces processing time. XML enables much better management, integration and synchronization of multiple data sources representing the same underlying data.

Real-Time Service: Requiring similar levels of integration as GSTP, Real-Time Service has an additional requirement for real-time integration and derived decision making. As many enterprises are attempting to reduce costs and risks through the automation of financial business operations, the ability to provide real-time statements of cash and exposure is becoming a key requirement for financial service providers (FSPs). XML can provide the framework within which FSPs may integrate systems and support their customers' business needs.

XML HANDLING IN APPLICATIONS—BOTTLENECKS

Today, many applications that use and manipulate XML do much of their XML processing in the application layer. Much of this is done using a lot of custom code, and typically delivers less than optimal performance. Although it is quicker to develop this way, the approach is often riddled with some of the bottlenecks mentioned below.



[FIGURE 1] Typical XML processing at application layer

Parsing XML: XML parsing is one of the most common and memory-intensive operations that occur when dealing with XML. SAX and DOM are the commonly used API (Application Programming Interface) paradigms. Both of these models apply a “use and throw” approach to XML parsing. There is a requirement to store the parsed document and create an efficient scheme to query it because: (a) document size has a direct impact on the efficiency of parsing and memory consumption, and (b) there is an inherent need to query XML data repeatedly once it is parsed.

Querying XML: As more and more business-critical data is in the form of XML, the need to query the information represented by the XML data becomes inevitable. Standards for querying XML data using W3C-recommended query languages like XPath and XQuery are gaining acceptance quickly. To efficiently query the XML data, a good query engine that implements XML operators and generates an optimal query plan is important. Equally important is an efficient and fast indexing scheme for XML data.

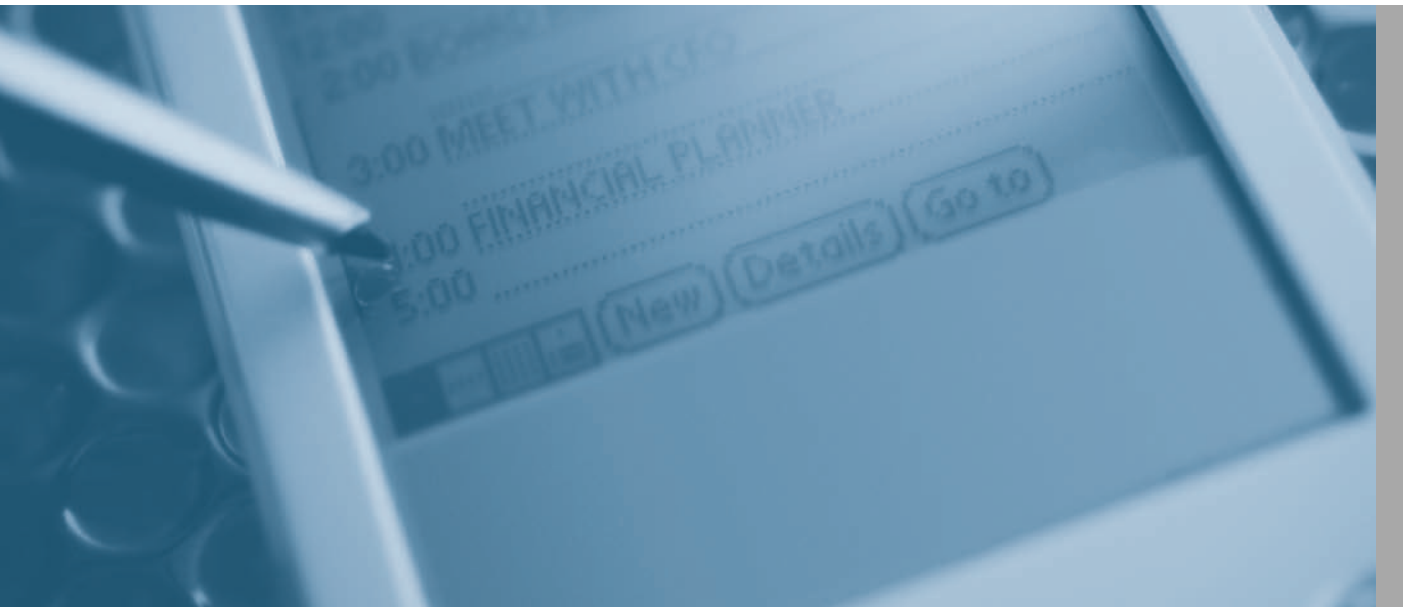
Transforming XML: XSL transforms are used for converting documents from one XML format (or standard) into another XML format (or standard), or into any other desired format. The transformation is achieved using XSLT processor; however, XSLT processing is computationally expensive.

Storing XML: XML documents can be archived either in file systems or in a database. While storing them in a file system is simpler, it is not practical for enterprise applications. Database management systems are the optimal way to store them, as the XML documents have to be used subsequently for many purposes, including querying.

The XML Management Package makes application development quick and cost-effective, enabling seamless integration of applications.

ROLE OF DATABASES IN MANAGING XML

The above discussion clearly illustrates that RDBMSs must be able to manage relational data as well as the business-critical XML data. So how would one bridge the gap between the document-centric world of XML and set-oriented world of relational databases? Does one store the XML documents in the database or as files in the file system? Does one shred the XML documents into relational data or store them “as is” natively? What is the impact of querying in each of these approaches? What are the other trade-offs? This section details the different options and tries to answer some of these questions. The rest of this document highlights how a user can take both approaches simultaneously using the XML Management Package.





XML STORAGE AND INDEXING

As organizations increasingly adopt XML and technologies built on XML, storing XML data quickly becomes critical. Although different vendors with varied backgrounds and strengths may promote their favourite way of storing XML data, there are essentially three ways of doing it.

1. Extract (shred) XML data and store it as relational data in an RDBMS

In this approach the XML document is parsed and the data contained in the document is shredded and stored as relational data in a relational database.

The advantage to this approach is that it makes use of the existing database system for dealing with XML data. In many cases, these are relational database systems that are well-tuned and highly efficient.

The disadvantage is that any change in access pattern also typically warrants a change in the database schema, which is not a scalable approach. The document (white space) fidelity will be lost and reconstructing the original document back would be rather difficult and cumbersome.

This approach is suitable for applications where XML messages (e.g., financial transactions), upon arrival at the receiving side, are represented as FIXML or FpML data, and end up in a relational database.

2. Store XML inside a database that supports integrated XML data management

In this approach the XML document is stored in a data store that interprets and preserves the structure and fidelity of the XML data.

Because this approach (and the underlying system) manages XML documents much differently from relational data, it uses the most optimal methods for storing, indexing and querying the XML data, rather than relying on relational storage, indexing, or querying techniques.

There are many specialty XML servers in the market that use this approach and do a good job. One drawback is that they are distinct (in terms of storage and processing) from the RDBMS that already exist in an enterprise, making it an additional IT software infrastructure that must be administered, maintained, and integrated with existing systems. Sybase supports integrated XML management directly into their RDBMS system, relieving the drawbacks mentioned above. This

approach is suitable for applications in which XML documents (e.g., an accident report, a mortgage application, or an insurance claim) presented to the system by the application must be maintained intact for, say, legal and regulatory purposes.

3. Store XML as files

In this approach the XML document is stored as raw text, typically as a file in an operating system file store. Though this approach is very easy to use, it doesn't typically provide transactional semantics or rich querying capabilities. Nevertheless, this may be the easiest path for XML adoption for many users. However, in a number of real world scenarios, a "hybrid storage" approach is more suitable when some of the business-critical data is stored in a relational table and the supporting data is stored "as XML documents," BLOB, in a relational database or files that can be accessed through a relational database.

Almost all organizations depend on RDBMSs for business-critical transactional data and would expect that system to either support XML data management directly or integrate well with a specialty XML data store. Different users have different storage and query needs. Sybase ASE, true to its commitment to providing flexibility to users and maintaining openness, supports all these methods of storing the XML data.

XML QUERY

Like unique and distinct data storage needs imposed by XML data, interesting querying needs are also imposed by XML data. Some typical query patterns involving XML are:

1. Extracting certain facts from XML data

Once a user stores the XML data in a DBMS, whether the data is stored in shredded form (as relational tables) or in native form (as an XML document), support for extracting facts will be important. For example: "How many trades were made on a ticker symbol where the ticker price went below a particular value and the average volume is higher than a certain amount?" OR "Give me all the suppliers of the company who are also consumers of the company and have purchased products in the last three quarters."

2. Searching an XML document

In this type of query, a user may be interested in finding a handle to all the documents where a certain pattern appears. For example: "Give me all mortgage applications that have an amount greater than \$300,000 as the value of the property but not as the loan amount". This query is fairly simple, but others could be more complex and require a full-fledged search of an XML document where the system tries to find all the documents that are most relevant to a user's criteria.

A minor variation involves searching an XML document to locate a particular document for further processing. The user would typically provide an ID or some other matching criteria to get at the document of interest. For example: "Give me the mortgage application for loan application #123456." This is a typical RDBMS query with the loan application number being the primary lookup key, and the column is indexed to optimize the query performance.

3. Aggregation

Another common query need involves aggregating the data from different XML documents. This is similar to the aggregation and grouping done in SQL queries. But it could be much more sophisticated because of the hierarchical nature of data. For example: “Give me the address of a customer from the address book XML document and combine it with all his recent orders to create an invoice that contains the customer’s information from the address book and his recent transactions from all the order XML documents.”

Some of these usages are more natural for traditional relational databases and some of them are more specific to the integrated XML storage. The heart of any database is the query optimization and query execution engine. Relational algebra, which has been around a few decades, and the query engines built around that concept, deliver exceptional query performance for relational data access. The XML data access needs a different query algebra designed to perform operations on XML data using the W3C-recommended XPath, XQuery, etc. Sybase has played a key role in the creation of the ANSI SQL standards and continues to be a staunch supporter of standards in the areas of SQL and XML querying. Sybase ASE supports both traditional SQL access to relational data and key XML querying standards such as SQL/XML (SQLX), XPath, and XQuery. They provide a simple yet powerful standards-based way of querying the data stored in ASE.

Sybase ASE supports query engines that are tailored to perform the SQL and XML operations. ASE leverages the “fast XML indexes” technology to provide high query performance. At the user level, it provides a more integrated approach to querying XML, where syntax is based on SQL, a familiar query language to the vast majority of developers who are accustomed to using an RDBMS.

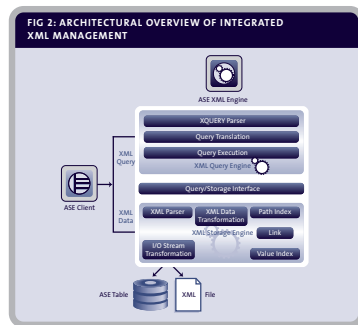
XML MAPPING AND TRANSFORMATION

Whether the XML data is stored natively as an XML document or stored in an extracted (shredded) form, there is a need to either: (a) extract a portion of the XML data matching a particular criteria, or (b) assemble XML data from different data residing in, for example, a relational store. This activity is nothing but transformation of either SQL or XML data into an XML format desired by the application. Hence, the ability to perform such XML transformations is a key requirement for XML data management systems.

Sybase ASE has strong support for XML mapping and transformation. ASE has added an extension to Transact SQL to directly produce the XML output from the stored relational data. It also supports W3C standards like XPath, XQuery, as well as ANSI standards like SQLX, all of which readily help in the transformation. Also, using the Java support in ASE, one can deploy their favourite XSLT engine within ASE and transform XML information into any other format. In all these cases, ASE nicely leverages the internal query and indexing capabilities and, most importantly, takes a modular and layered approach (where new capabilities can be quickly designed or enabled on top of the existing ones) to greatly improve the performance and scalability of the transformations.

OVERVIEW OF XML DATA MANAGEMENT SUPPORT IN ASE

Sybase ASE provides integrated XML data management capabilities built around the mature, dependable ASE code base. This provides the XML functionality on top of the core ASE database functionalities that enterprises rely on, such as distributed transaction and querying, scalability, high availability, robustness, and high performance. We have leveraged the years of experience in designing databases to build a next-generation data management system that deals with relational and XML data and delivers an enterprise-class system that meets users' needs.



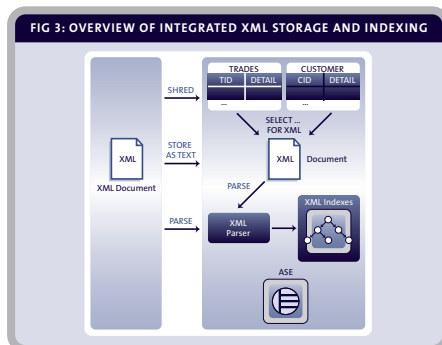
[FIGURE 2] Architectural Overview of Integrated XML Data Management

INTEGRATED XML STORAGE

ASE natively stores the XML data and lets the user manage it with the same ease with which users have been managing the relational (SQL) data. With integrated XML storage, users no longer need to shred their XML data to make it fit into a relational data storage. This includes indexing, backup, replication, and a number of other data management functionalities. By integrating the XML storage and querying functionality natively into the ASE engine, Sybase readily lets the user take advantage of ASE's XML capabilities. See Figure 3 for an illustration.

XML INDEXING

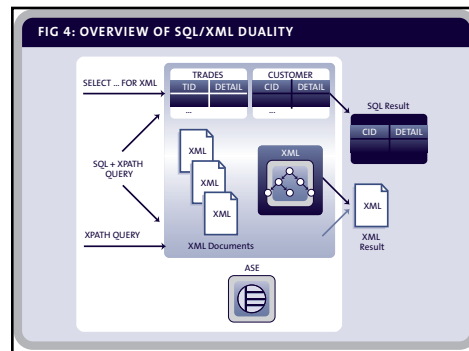
ASE provides a patent-pending Fast Index for XML that indexes any XML element, attribute, or path, enabling high query and search performance. This automatic index creation and maintenance functionality is schema-independent and relieves users from having to do these upon catalog creation. See Figure 3 for an illustration.



[FIGURE 3] Overview of Integrated XML Storage and Indexing

SQL/XML INTERCHANGEABILITY

The ASE XML Engine provides complete XML-SQL interchangeability, allowing users to perform XML operations on SQL data and SQL operations on XML data. This minimizes the need for special XML tools for retrieving the XML data out of the database. See Figure 4 for an illustration.



[FIGURE 4] Overview of SQL/XML Duality

XML MAPPING AND TRANSFORMATION

ASE 12.5.1 provides a competitive set of features to map and transform SQL and XML data. Using these features, users can readily (right out of the box) transform SQL result sets into XML, which can then be consumed by various applications in their enterprise. In many cases, this process can be accomplished by simply adding the words “FOR XML” at the end of a SQL select statement. Sybase has also partnered with HiT Software, provider of the SQL/XML mapping tool JAllora, for richer and more efficient mapping of XML and SQL data.

OPEN STANDARDS (ANSI SQL, W3C) SUPPORT

Sybase has played a key role in defining ANSI SQL standards, and continues to be a staunch supporter of standards in the areas of SQL and XML querying. Sybase ASE supports both the traditional SQL access to relational data and the key XML querying standards such as SQL/XML (SQLX), XPath, and XQuery. ASE 12.5.1 implements a significant subset of the XQuery language and is upwards compatible with the XPath language, which was the basis of the XQL processor provided in ASE 12.5. The rich set of constructs supported by ASE range from granular to complex query and data retrieval operations, including wild card expressions. ASE 12.5.1 constructs also provide a smooth migration path for current applications that use XML functionality provided in ASE 12.5.

The following table recapitulates the key features and their technical benefits. Subsequent sections highlight typical use cases and application areas of these features.

FEATURE	BENEFITS
Integrated XML Support	Store and retrieve well-formed XML data Schema independent
SQL and XML Duality	Complete interoperability between SQL and XML at language and storage level
XML Indexing	<ul style="list-style-type: none"> ■ Renders high query performance ■ Self defined indexes — Provides ease of management
XPath and XQuery Support	<ul style="list-style-type: none"> ■ Granular results — Returns documents, document fragments, elements ■ Wild card support for complex queries ■ Functions to process data in XPATH: concat, toupper, tolower, normalize-space etc.
XML Views/Mapping	Maps SQL data as XML, and XML as SQL data
XML Transformations	Allows XML transformations to be applied on both the stored XML document and the XML view

USAGE SCENARIOS

Technology analysts, industry regulators, and industry consortia all see XML and Web services as key technologies in meeting the integration challenge. The benefits of moving to XML and Web services include greatly reduced development, deployment, and ongoing maintenance costs, as well as an ability to respond much more quickly to changing business requirements. The size and scale of these benefits are dramatic.

INSURANCE INDUSTRY—VALUABLE INTEGRATED XML SUPPORT (STORAGE AND QUERYING)

In the insurance industry there is a strict requirement to store documents such as mortgage applications and insurance claims, “as is”— that is, without decomposing the document into various forms. One of the reasons is to maintain a document of evidence for auditing and future processing purposes. Companies in the insurance industry are consolidating the information exchange using the XML format. Evidently all the details that get exchanged and stored are in the form of XML.

This is an area where the integrated XML storage and query capabilities of XPath/XQuery are very valuable in handling the XML data as mandated by the industry leaders. Sybase ASE 12.5.1 provides state-of-the-art support for integrated XML management. Here is a quick look at how the incoming document can be stored and parsed (either in realtime or in a deferred mode). In this example, a loan document in XML format is first parsed and stored as a XML document in a table, and later queried using ASE XML management capabilities.

The built-in `xmlparse` function parses the XML document passed as parameter, and returns an image value that contains a parsed form of the document. In the examples below, the `xmlparse` statement is used to achieve several things: (a) parse the document to ensure the conformance to the XML syntax, (b) generate the Fast XML Index on the entire XML document to help speed up the query performance, and (c) store the parsed XML document in the underlying table.

Figure 1

```
INSERT INTO mortgage_app_table (mortgage_app_id, mortgage_app_doc)
VALUES (1032423, xmlparse (@loan_doc))
UPDATE mortgage_app_table
SET mortgage_app_doc = xmlparse (@new_loan_doc)
WHERE claim_id = 1032423;
```

Once the mortgage applications are stored natively as XML, there will be an eventual business need to query the key information from the XML document and perhaps use that data along with other details in the relational database. This can be achieved using the following query that queries the table where mortgage applications are stored natively. From that table it extracts the loan amount requested from all the loan applications for fixed loans with conventional mortgage type.

Figure 2

```
SELECT mortgage_app_id, xmlextract
('/LoanApplication/LoanInfo [MortgageType = "Conventional" and AmortizationType =
"Fixed"]/amount',
mortgage_app_doc)
FROM mortgage_app_table
```

Above is a simple example that illustrates the language constructs. One can apply the xmlextract function to the parsed XML documents in the image column in the same way it is applied to the unparsed XML documents in the text column. Operations on parsed XML documents will generally execute faster than on unparsed XML documents.



The following example illustrates how the powerful query features such as descendants and wild-cards can be used to query the XML document stored natively, using the XPath/XQuery language.

Here we are extracting the assets and liabilities in actual amounts from one of the target loan applications qualified by social security number. Note the applicant with this social security number could either be a borrower or a co-borrower. The qualification is done using `xmltest`, which is a convenience function, similar to the `LIKE` clause of `SQL` for predicate matching. Here the `'//'` is a descendant which lets the `xmlextract` and `xmltest` expressions search anywhere in the document.

Figure 3

```
SELECT
xmlextract ('//assets_and_liabilities//assets/*/amount', mortgage_app_doc),
xmlextract ('//assets_and_liabilities//liabilities/*/amount', mortgage_app_doc)
FROM mortgage_app_table
WHERE '//*[SocialSecurityNumber = "123-45-6789"]' xmltest mortgage_app_doc
```

FINANCIAL SERVICES INDUSTRY—FLEXIBLE AND EASY XML TRANSFORMATION

The `FIX` protocol and its variation `FIXML` are very popular in the financial services industry. `FIXML` is an `XML` extension of the `FIX` protocol. `FIX` handles real-time electronic exchange of securities transactions, both pre-trade and trade execution for equities and fixed-income instruments. `FIX` is very flexible and is heavily customized by users at the application message level. `FIXML` was designed to minimize the changes required in existing `FIX`-based systems. It takes a `FIX` tag value format and represents it in `XML`. These `FIXML` messages are then embedded within the traditional `FIX` headers and trailers, meaning that only the addition of an `XML` parser is required for an existing `FIX` engine to communicate using `FIXML`.

As more and more applications start using `FIX` and `FIXML`, there will be a need to transform the `SQL` data stored in the database into `XML` output. In almost all cases, this is done today at the application layer as a multi-step process, as illustrated in one of the previous sections. This can be avoided by simply adding a `FOR XML` construct at the end of a regular `SELECT` statement to generate the results in `XML` format instead of relational format. Here is an example.

Figure 4

```
SELECT ticker, price, quantity, account from trade_table
WHERE trade_id = "123FG45C"
FOR XML
GO
<resultset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<row>
<ticker>SY</ticker>
<price>15.50</price>
<quantity>5000</quantity>
<account>123-45-6789</account>
</row>
</resultset>
```

The FOR XML construct in ASE 12.5.1 supports multiple options that can help in generating customized mappings and transformations of the SQL data into XML data, which can easily be exchanged between applications.

A counterpart to the above example is the consumption of XML data (e.g., a FIXML message) that arrives at the database, and the decomposition of that data into relational storage. Note the users have the flexibility of shredding, storing the data “as is,” or performing a combination of the two to exploit the full range of storage and querying capabilities provided in ASE 12.5.1 for dealing with XML.

Typical financial services applications demand a very high transaction rate compared to the insurance industry use case that we discussed earlier. In such cases, every additional step in the transaction path would reduce the throughput. This means that due to transaction performance requirements, there may be applications for which shredding or parsing and index generation would not be possible when storing documents. To address this, ASE provides a way for users to store the XML message “as is” (e.g., as text data), and provides on-demand XML index generation when the XML data is queried. ASE also provides flexible ways outside the transaction path to shred the XML data and populate tables with the relevant trade data in relational form.

AREAS OF APPLICATION

ENTERPRISE INFORMATION INTEGRATION

The Sybase Adaptive Server family provides the ability to integrate enterprise information systems through the database extensibility mechanism of Component Integration Services (CIS). CIS provides intelligent distributed query-processing capabilities for remote data access, and also provides the enabling technology for specialty data stores. The CIS functionality, in combination with the Sybase EnterpriseConnect product suite, augments the heterogeneous data access and data movement functionality of ASE. It also extends the reach of ASE to enterprise-wide legacy systems, as well as to object and non-relational systems. These extensibility mechanisms enable Sybase Adaptive Server to co-exist in a heterogeneous world and simplify the task of application development by presenting a uniform and consistent view of enterprise systems. The addition of Web services support in ASE and the trend of enterprises to more openly embrace Web services open the door to a whole new class of applications that can leverage Sybase Adaptive Server for information integration.

WEB SERVICES

Web services technology is changing the Internet by enabling application-to-application interaction over the Web, regardless of the language, platform, or data format. It provides a language-neutral, environment-neutral programming model that accelerates application and data integration. This transformation is being fueled by the automatic interaction at the software level between businesses. XML messaging is a key and fundamental building block for Web services. The use of XML as a platform-independent data exchange format plays an important role in a number of financial services industry initiatives. These include:

- A standard way of sending and receiving the information to complete the business processing (XML, with an additional standardized envelope called SOAP, which is also part of the ebXML standards).
- A standard way of publishing descriptions of the conversations that a Web service can be involved in. The leading standards are UDDI (which can provide WSDL as a way of describing actual XML document exchanges within the conversation) and the forthcoming ebXML registry.

Sybase recognizes that Web services provide architecture not just for integrating applications, but also for integrating data. ASE, in the 12.5.1 release, provides the capability both for managing data and for intelligent, optimized access to data. There are two distinct and important Web services functions supported by ASE.

The first is Web service enablement of existing logic within ASE by describing the interfaces for that logic in WSDL and providing the capability to send and receive SOAP attachments. This allows new applications to be written using exposed Web services.

The second is Web service integration, where ASE mediates a call to one or more Web services and uses the results from the Web services along with relational and/or XML data stored within ASE.

The initial support of Web services in ASE enables it to act as both a consumer and producer in an enterprise-level Web services architecture. It is a precursor to a richer integration of Web services in ASE that virtualizes and globalizes database schemas, tables, and their operations.

CONTENT MANAGEMENT

Today, business applications need a composite view of all information sources, whether structured, semi-structured, or unstructured. The resulting content model must provide the combined capabilities of content management systems and traditional databases: document based, fast query and search capabilities, extensive indexing, and application development support. XML adds structure and meaning to document content, making it easier to index and find content.

Sybase Adaptive Server Enterprise provides:

- Fully integrated XML support, storing XML content “as is,” and providing true XML retrieval capabilities based on XPath/XQuery standards
- Enhanced, full-text search capabilities with awareness of XML element structures
- Extended file system support that exposes the content of the file system as a relational model for manipulation using SQL
- Component Integration Services and Enterprise Data Access Services that provide a comprehensive set of data integration functionality
- Web services support that helps integrate non-relational content expressed as Web services in ASE with data from heterogeneous data sources
- Rich transformation support

Sybase ASE bridges the gap between unstructured and structured content sources by creating an XML-based content model that is self-describing, is automatically indexed for efficient retrieval, and provides unified access to all information sources.

SUMMARY

In the world of IT, XML is becoming the lingua franca for businesses exchanging information electronically. In addition, the technologies built on XML are becoming pervasive. Sybase's Integrated XML Engine in ASE 12.5.1 allows users to efficiently store, query, manage and exchange XML data in its native format, based on W3C, XML, and other open Internet standards. With ASE 12.5.1, which has integrated XML storage and query capabilities, Sybase provides a solid data management platform that exploits the potential of XML to enable smooth information interchange. It combines the power of a rich relational database technology with the potential of XML technology by fully absorbing the W3C XML data model into the ASE XML server. By integrating SQL and XML Data Management functionalities natively in the same server, ASE reduces the cost of ownership and the number of moving parts in a complex IT infrastructure. ASE provides a solid data management platform to exploit the power of XML to its full potential while leveraging existing IT investments.

SYBASE®

Sybase Incorporated
Worldwide Headquarters
One Sybase Drive
Dublin CA, 94568 USA
T 1.800.8.SYBASE
F 1.925.234.5678
www.sybase.com

Copyright © 2004 Sybase, Inc. All rights reserved. Unpublished rights reserved under U.S. copyright laws. Sybase and the Sybase logo are trademarks of Sybase, Inc. All other trademarks are property of their respective owners. ® indicates registration in the United States. Specifications are subject to change without notice. Printed in the U.S.A. LO1831 MIL1038