ORACLE DATABASE 12c

An Oracle White Paper
26 June 2013

Oracle XML DB in Oracle Database 12c

ORACLE

Executive Overview

XML is an extremely popular way to persist and exchange business-critical information. XML is an open standard managed by the W3C, under the control of no single vendor. Many industry segments have developed XML-based standards for representing information. These standards typically utilize XML Schema, a W3C standard for defining the structure and content of an XML File. XML-based standards can be found in healthcare, financial services, manufacturing, publishing, law enforcement and the public sector. XML also provides the foundation for SOAP based application development. In a growing number of situations, government regulation mandates the use of such standards when exchanging information. These trends have led to massive increases in the volume of XML that organizations needs to deal with forcing them to adopt platforms that manage XML with a degree of rigor and security similar to that used for other kinds of operational data.

To meet this need, Oracle developed Oracle XML DB. Oracle XML DB is a high-performance, native XML storage and retrieval technology that is delivered as a part of all versions of Oracle Database. Oracle XML DB allows an organization to manage its XML content using the same trusted technology that it uses to manage its relational data. This reduces costs and improves return on investment since a single platform is used to manage and secure all mission critical data. Oracle XML DB was first released with Oracle 9iR2, and it has been enhanced in each subsequent major release of the database.

Introduction to XML

XML has the advantage of being inherently self-describing and is both human and machine readable. The self-describing nature of XML makes it very good for representing dense data as well as sparse or extremely variable data. The XML data model is also highly extensible, allowing organizations to easily customize XML content models to meet specific information storage and retrieval requirements. The primary use cases for XML are

- Data capture: Storage of data generated by sensors and loggers
- Data Exchange: Information exchange between loosely coupled systems.
- XML Persistence: Persistence based on industry-standard data models
- XML Persistence: Persistence of application objects, metadata and state.
- XML Persistence: Persistence of content created using popular office productivity software.

These use cases are described in more detail below. In addition to the basic XML standard, the W3C has developed a number of related standards that are useful when working with XML content. These include the following:

XML Schema

As the complexity of XML documents increases it becomes necessary to have a standardized language to describe the contents of an XML document. The XML Schema standard defines approximately 49 primitive data types and a vocabulary that allows these primitive types to be combined together to construct other more complex objects. Using this vocabulary it is possible to create a set of element and type declarations that accurately and unambiguously describe the content and structure of an XML document. Part of the XML Schema standard is an XML Schema that defines the XML Schema language, known as the "Schema for Schemas". An XML Schema is simply an XML document which is compliant with the "Schema for Schemas".

XML Schema has been widely adopted because it allows organizations to accurately describe what information they need to exchange, and to validate that the information being exchanged conforms to the agreed specification. Many industry standards bodies have used XML Schema to define the persistence and exchange models needed to exchange information between customers, suppliers and partners.

XQuery and XPath

XQuery is the natural query language for XML content, in the same way that SQL is the natural query language for relational content. XQuery uses a superset of the W3C's XPath expression syntax to address specific parts of an XML document. It supplements this with a SQL-like "FLWOR expression" which is used to iterate over sets of nodes and to perform join operations. A FLWOR expression is constructed from the five clauses after which it is named: FOR, LET, WHERE, ORDER BY, RETURN.

XQuery can also be used to create new XML documents. The language defines element and attribute constructors that can be combined together in nested structures to synthesize the required XML document.

XQuery is based on the XQuery Data Model (XDM). In XDM all values are represented as a sequence of 0 or more items. The items in a sequence can be XML nodes or atomic values. Atomic values may be any of the primitive types defined by XML Schema. T full list of types is based on the primitive types defined by XML Schema.

Recent enhancements to the XQuery language include the XQuery-Update recommendation and the XQuery Full-Text recommendation. XQuery-Update makes it possible to modify the content of an XML document, supporting insert, delete, modify and rename operations. XQuery Full-Text provides support for performing complex full-text searches of XML content, with the ability to search for text at the document, fragment or node level.

SOAP

SOAP is another standard that is popular with application developers. The SOAP standard makes extensive use of XML to provide a service-based infrastructure A SOAP service is defined using a Web Services Description Language (WSDL) document. To invoke a SOAP service, a SOAP client sends a request document to the SOAP server. The results of invoking a service are returned to the SOAP client as a response document. XML is used for the request and response documents. The WSDL is also an XML document. It provides the location for the end-point of the service and uses an embedded XML Schema to define the structure of the request and response documents.

XML Use-cases

XML for Data Capture

Data generated by sensors and loggers, or by application logging, is stored as XML. In this scenario an extremely large volume of XML data is generated in a relatively short time period. The XML may be stored as a small number of very large files or a very large number of relatively small files. This data needs to be integrated into the business's application processes.

XML for Data Exchange

In this scenario, systems use XML messages to communicate with each other. XML is generated from (typically relational) data managed by one system, transported to some other location, and then ingested into the (typically relational) data stores managed by the other system. The use of XML as an exchange medium provides an abstraction layer that allows one application system to re-organize its data without impacting any applications that require access to that data.

In some models, such as SOAP based messaging, extremely large numbers of small (4Kb-100Kb) XML documents are exchanged in near real time. In other cases, small to medium volumes of large (100Kb to10+GB) XML files are created and processed using more traditional batch-generation and ETL-processing techniques. Often these messages are compliant with industry standard XML Schemas that have been developed by the relevant Industry standards bodies.

XML Persistence for Industry Standard XML Data-Models

One of the drivers for the growth in XML persistence has been the emergence of industry-based XML standards. As these standards evolve they define extremely complex and highly variable information models in order to meet the needs of the organizations they serve. Analysis has shown that some of these XML Schemas describe models that would translate into relational data models containing thousands of tables. As the degree of complexity and variability in the model increases, the cost and time required to develop software that performs a bi-directional, lossless translation between the relational model and the XML model become prohibitive.

Given recent improvements in XML querying and indexing techniques, organizations are trying to see if they can save money and resources by working directly with their XML content. This leads to a significant increase in the amount of XML which needs to persisted and managed in a professional manner. Significant volumes of XML persistence also arise from the requirement to preserve even simple XML messages for non-repudiation purposes.

XML for Application State Persistence

The flexibility inherent in the XML model makes it attractive as way of persisting data in applications where the data model is highly volatile or, in many cases, not known in advance. Adopting XML-based persistence makes it possible to create applications where the data model can be modified on the fly, either as part of the installation process, or in some cases at runtime. Many application developers use XML as a way to deliver extensibility in a way that does not require changes to the data storage model every time the application's requirements change.

Some application developers will choose to use a hybrid model, where the well known parts of the data model are persisted using conventional relational tables, and other, less well-defined or volatile parts of the data model are persisted using XML. Others will choose to use a document store model, where all of the application data and state is contained in a set of documents that are accessed using a set of simple, well known global identifiers (GUIDs). Finally there are the use-cases where a name-value pair storage model is persisted using XML. This is particularly common when an object-persistence layer, such as Hibernate is being used to map between the objects defined by the application and the objects understood by the storage layer.

This hybrid model is common in applications like content management systems. This kind of application has the requirement to allow the line-of-business user to define the classes of metadata that will be associated with the objects the application manages. This process takes place dynamically once the application has been deployed. Once the metadata has been defined, the application expects the storage infrastructure to treat it as a first class citizen with respect to providing efficient data management and query capabilities.

XML for Office Productivity and Technical Authoring Software.

The flexibility and variability inherent in the XML data model makes it an attractive mechanism for storing content generated using office productivity suites like Microsoft Office and technical authoring software such as Frame and ArborText. This is not surprising, given that the precursor of XML was SGML, a standard designed explicitly for this purpose. The recent drive towards providing a greater degree of interoperability between the various vendors office productivity software has resulted in the creation of open source, XML-based standards for storing this kind of content. Most office productivity suite vendors are now shipping products that support the new XML-based content models, so there is a significant growth in the volume of this kind of content. The DOCX, XSLX and PPTX file formats used by Microsoft Office 2007 and later are examples of this trend. These 'X' file formats are actually ZIP archives where the content of the office document is stored as a set of XML files compliant with the Office Open XML standard.

The adoption of XML as the primary method of persisting office productivity documents is leading to a growing interest in XML-based content management (CM) systems. These systems differentiate themselves from traditional CM solutions by being able to understand both the metadata and the content of the documents that they manage, allowing their users to make much more effective use of the large volumes of information that was trapped in the proprietary document formats used by the previous generations of office productivity software.

Introduction to Oracle XML DB

Oracle XML DB is a high-performance, native XML storage and retrieval technology that is delivered as a part of all versions of Oracle Database. Oracle XML DB provides full support for all of the key XML standards, including XML, XML Namespaces, DOM, XQuery, SQL/XML and XSLT. By providing full support for XML standards, Oracle XML DB enables XML-centric application development. Application developers are able to use common XML techniques to store, manage, organize, and manipulate XML content stored in the database. Oracle XML DB also supports the SQL/XML standard, which allows SQL-centric development techniques to be used to publish XML directly from relational data stored in Oracle Database 12c.

Key Features of XML DB include

- Efficient XML storage
- Standards compliant query and update operations
- Powerful and flexible indexing
- Tight integration with Oracle's security and data management capabilities
- Support for XML-centric, SQL-centric and document-centric development
- XML and SQL interoperability

Efficient XML Storage

XML content is stored in the database using the XMLType data type. The XMLType data type is an abstraction that supports multiple storage models, allowing Oracle XML DB to optimize XML storage for the entire set of widely divergent usage models outlined in the first section of this whitepaper. Broadly speaking, Oracle Database 12c provides support for three primary storage models:

- Binary XML storage
- Object Relational XML storage
- Relational Storage

The fact that XMLType is an abstraction over the underlying storage model allows application developers to code their application in a way that is 100% independent of the way in which the XML is stored on disc. Earlier versions of the Oracle Database supported a CLOB based storage model for XMLType, but this is deprecated in favor of Binary XML storage starting with Oracle Database 12c.

Binary XML Storage

Binary XML stores XML in the database using a native binary representation of the XML. This format is known to as "Compact, Schema-Aware XML" (CSX). Using the CSX format reduces storage requirements by approxamtely 50%. The first step in CSX encoding is to tokenize all of the tags. This optimization is applied to both schema-based and non-schema-based XML.

For schema-Based XML, further storage optimizations are achieved by representing non-text values using native data types and eliminating unnecessary token information from the CSX format. This allows schema-based binary XML to deliver a total reduction in storage in the range of 60-75%.

Binary XML is stored on disc using Oracle's SecureFiles infrastructure. This delivers the maximum possible throughput for storage and retrieval operations on XML documents. Binary XML also leverages the SecureFiles sliding insert feature, allowing efficient, node-level, insert, update and delete operations on XML content. If further storage savings are desired, both schema-based and non-schema-based binary XML can be further compressed using the compression features of Oracle Database.

Wherever possible, operations on binary XML are performed using streaming techniques. This improves performance by avoiding the memory and CPU overhead associated with traditional DOM-based processing of XML content. Streaming is supported during ingestion for both parsing and encoding of XML content as well as for the validation of Schema-based XML. The CSX format is also optimized for streaming XPath operations. This allows the database to efficiently evaluate complex XPath expressions against large volumes of XML content. Multiple leaf nodes or fragments can be extracted in a single pass of the XML document.

The CSX format provides very flexible support for schema-based XML content. Documents associated with different XML Schemas, or different versions of an XML Schema can be stored in the same table or column.

Object-Relational XML Storage

Object-Relational storage persists an XML document as a set of SQL objects. The SQL object model is generated by compiling the object model defined by an XML Schema. This process is referred to as XML schema registration. These objects allow an XML document that conforms to the XML Schema, to be stored in the database with no loss of fidelity. Repeating elements in the XML Schema are automatically compiled into ordered collections, and recursive structures are handled automatically during the XML Schema compilation process.

Like binary XML storage, object-relational storage delivers significant space savings. The explicit mapping between the contents of the XML document and the SQL object model allow all tagging information to be discarded. Savings also arise by storing all non-text values as native Oracle data types.

Once the XML data has been ingested and stored as a set of objects, any operations performed on it are rewritten into operations on the underlying objects. XQuery and XPath-based operations on the XML data are compiled into the same algebra that is used for operations on relational content, allowing the Oracle optimizer to optimize XQuery in the same manner that it optimizes SQL.

Object-relational storage supports the concept of XML Schema extension, which is the W3C's preferred way of addressing changes to an XML Schema. Once an extension schema has been developed, it can be registered with the database. XML DB will modify the object model derived from original XML Schema to accomodate the types defined by the extension schema, during the schema registration process.

Oracle XML DB also support in-place schema evolution, which allows a defined subset of changes to be made to registered XML Schema without having to unload and reload any instance documents that are already stored in the database. In-place schema evolution is supported as long as the required changes mean that any existing corpus of instance documents is still valid with respect to the new XML Schema.

Relational XML Storage

With Oracle XML DB it is possible to use SQL/XML and XQuery to create views which expose relational data as XMLType values. This data can be queried using XQuery and XPath in exactly the same manner as XML data that makes use of binary or object-relational storage. In most cases, updates to these views can be handled using Instead-Of triggers.

Standards based XML Query and Update

All operations on XML data stored in the database are performed using XQuery and XPath. Oracle XML DB provides comprehensive support for using XQuery to manipulate XMLType values stored in Oracle Database. The SQL/XML extension to the SQL standard defines a set of operators, XMLQuery, XMLTable and XMLExists, that allow XQuery operations to be executed in the context of a SQL statement. This allows XML operations to use the same transaction semantics as other database operations. The SQL/XML extension also defines the XMLCast operator, which allows translation between the primitive types defined by XML Schema and the scalar types supported by SQL. New in Oracle Database 12c is the ability to update XML content using the W3C XQuery Update recommendation and to perform searches over XML content using the XQuery Full-Text recommendation.

Oracle XML DB also provides support for XQJ, also known as JSR-225. This is a Java standard that provides a JDBC-like interface for executing XQuery operations. This allows Java programmers to develop pure XQuery-based applications that work directly with XML data stored in Oracle Database . Oracle XML DB also provides a SOAP endpoint that allows the database to be a direct participant in SOAP based environments. This endpoint supports an XQuery service, a SQL service, and a mechanism for invoking PL/SQL packages, procedures and functions as a service.

Oracle XML DB also provides DOM-level access and manipulation of XML content via the DBMS_XMLDOM and DBMS_XMLPARSER packages. XSL transformation of XML content is supported via the SQL operator XMLTransform and the DBMS_XSLTRANSFORM PL/SQL package.

Details of how to use these features can be found in Oracle XML DB Developer's Guide.

XML Indexing

Oracle XML DB supports a number of different XML indexing models, which make it possible to efficiently index all kinds of XML content.

Oracle's binary XML format is designed to enable efficient indexing of XML content. The binary XML format was designed from the ground up to allow optimization of XQuery operations which result in

fragment and leaf level access to XML content. There are three distinct indexing techniques for XML storage.

- Unstructured XML Index
- Structured XML Index
- XML Full-Text Index.

Unstructured XML Index is primarily designed for use cases where neither the structure of the XML or the nature of the queries is known in advance. In its most basic form, it indexes every node in the document, allowing it to be used to optimize any possible query against the corpus of indexed XML documents. Maintaining a complete index can consume significant amounts of resources, particular for large volumes of XML. Oracle XML DB allows unstructured XML Index resource utilization to be controlled by restricting the index to a subset of the nodes in the document, or by running the index in an asynchronous manner.

Structured XML Index is focused on use cases where both the structure of the XML data and the set of queries that will be executed are well known. Behind the scenes, structured XML Index projects the data required to answer the queries into a set of relational tables. When an XQuery expression is executed, the XQuery processor executes a relational query over these tables to determine which XML documents satisfy the XQuery expression. XQuery expressions that involve leaf level extraction, can also be optimized by extracting data directly from the index.

The XML full-text index is new in Oracle Database 12c. It optimizes XQuery Full-Text operations on XML documents stored in the Oracle Database. It will be discussed in detail later, in the What's New section of this document.

Indexing object-relational storage is similar to indexing any complex relational hierarchy. Oracle automatically creates the B-Tree indexes need to optimize bi-directional traversal of the object hierarchy. DBA's are free to add additional B-Tree indexes to support other access paths.

XML Optimized Content Repository

Oracle XML DB incorporates a high-performance XML-centric repository. This repository makes it possible to organize and access content stored in Oracle Database using a familiar file/folder metaphor. The repository can be accessed directly from SQL or by using standard Internet protocols such as HTTP, FTP and WebDAV. This makes it possible to access XML content stored in the database directly from common desktop tools such as Microsoft Office and Windows Explorer.

The repository provides support for basic content management operations such as versioning with check-out and check-in capabilities and an event model that makes it possible to react to operations on folders and files in the same way that database triggers make it possible to react to operations on relational tables.

The Oracle XML DB Repository allows Oracle Database 12c to meet the needs of content-centric, as well as data-centric XML application development.

Oracle XML DB enhancements in Oracle Database 12c

With the release of Oracle Database 12c, Oracle extends its industry-leading XML support, ensuring that Oracle remains the best platform for storing, managing, and querying all types of XML content. Oracle Database 12c delivers on Oracle's ongoing commitment to deliver new functionality as well as improved performance and scalability with each new release. Starting with Oracle Database 12c, Oracle XML DB and the Oracle XML DB repository are now mandatory features of the Oracle Database. It is no longer necessary to worry about how to install Oracle XML DB or whether or not Oracle XML DB is available in a particular database instance. New functionality delivered with Oracle Database 12c includes support for the following:

- XQuery Update.
- XQuery Full-Text.
- Oracle Multitenant option
- HTTPS and Oracle EM Express
- Digest Authentication

Performance, Availability and Scalability improvements for XML processing in Oracle Database 12 include the following:

- XMLTable optimization
- Partitioning and Parallel processing
- Integration with Logical Standby and Oracle Golden-Gate
- Support for Rolling Upgrades

Ease of use improvements includes the following:

- XML Manageability Packages

XQuery Update

In Oracle Database 12c, Oracle XML DB adds support for the W3C XQuery update recommendation. XQuery Update enables the following operations on an XML document :

- Insertion of a node.
- Deletion of a node.
- Modification of a node
- Rename of a node

In Oracle Database 12c XQuery update operations are invoked using the XMLQuery operator. The operator can be used as part of a SQL SELECT or UPDATE statement. In the case of an UPDATE statement, standard database transaction semantics will apply. Where possible, the XQuery-Update operation will be translated into a partial update of the target XML document.

Oracle XML DB supports the 'copy modify' form of XQuery Update. A simple example of this is shown below:

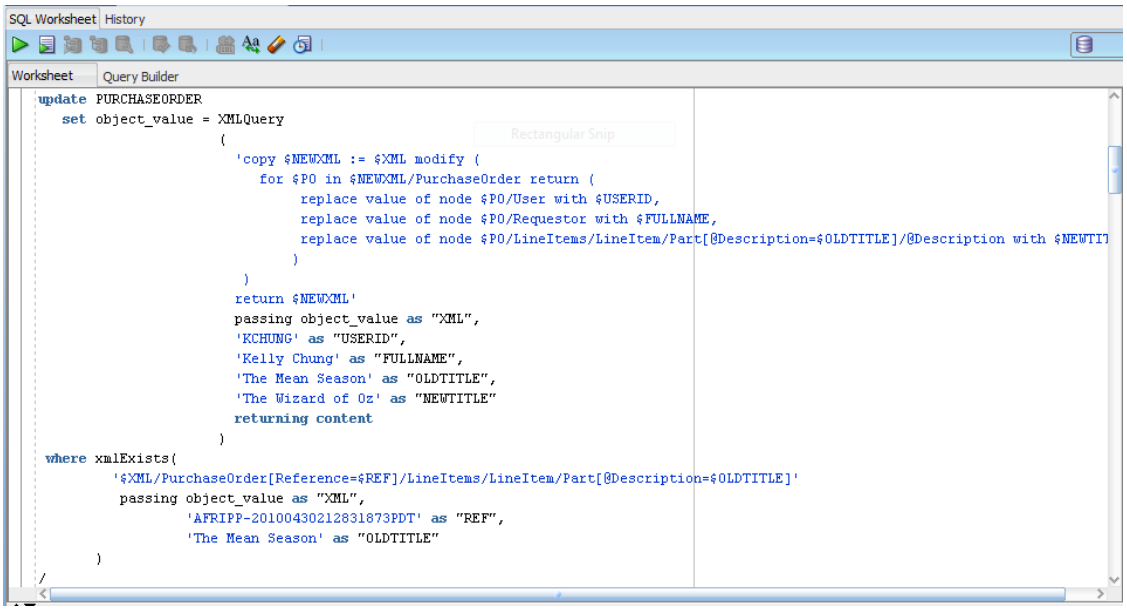

Figure 1. XQuery Update operation as part of SQL UPDATE statement

Oracle's proprietary operators for updating XML content, including UpdateXML, InsertChildXML, DeleteXML, AppendChildXML and InsertXMLBefore are deprecated in Oracle Database 12c. The XQuery-Update facility provides a comprehensive, standards-compliant mechanism for updating XML content stored in the Oracle Database, which totally subsumes the functionality provided by the deprecated operators.

Note that XQuery-Update support is also enabled in Oracle Database 11g starting with patch set 11.2.0.3.0. Full details of the XQuery-Update recommendation can be found at http://www.w3.org/TR/xquery-update-10/.

XQuery Full text and XML Full text Indexing

XML can be used to represent many different kinds of information. The nature of this information spans the gamut from highly structured data (fixed schemas, known types such as numbers, dates) through semi-structured data (flexible schemas and types) to markup data (text with embedded tags) and unstructured data (untagged free-flowing text). As the XML data becomes less structured the need to be able to search using Information Retrieval (IR) techniques grows.

Full text searches are very different from substring searches. A full text search searches for tokens and phrases rather than substrings. For instance, a substring search for items that contain the string "sport" will return an item that contains "transport". A full text search for the token "sport" will not. The XQuery 1.0 recommendation defines an operator called "contains" that provides substring matching.

The W3C XQuery and XPath Full Text Facility 1.0 (XQuery-FT) extends the syntax and semantics of XQuery 1.0 and XPath 2.0 to provide full text searching capabilities

Full text search needs to support language-based searches. An example of a language-based search is "find all items that contain a token with the same linguistic stem as 'sport'" (finds "sport" and "sports" but not "transport") or "find all items that contain the tokens "oracle" and "database" within 3 tokens of each other." Both of these searches are impossible to implement using a simple substring approach. It should also be noted that full text searching is typically case insensitive where as substring based searching is typically case sensitive.

One other challenge with full-text searching is ordering or ranking of results. Full text search is not an exact science. Search results need to be ranked to show them in an order which will place the most relevant results first.

In Oracle Database 12c, Oracle XML DB extends its XQuery implementation to provide support for XQuery Full text. Oracle is the first mainstream database vendor to provide support for this important W3C standard. An example of a simple XQuery-FT search is shown below:

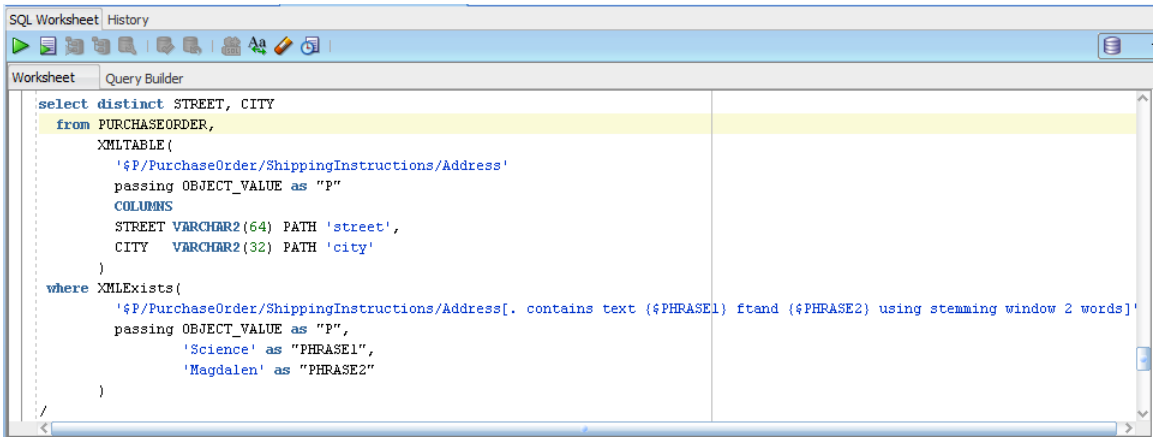

Figure 2. XQuery Full Text search

XQuery Full text optimizes full text searching by taking advantage of the capabilities of Oracle Database's Oracle Text feature. Oracle Text is a well established feature of Oracle Database that provides powerful full text search facilities through the SQL CONTAINS operator. The key to Oracle Text performance is the "ctxsys.context" index, which allows the database to efficiently index and search extremely large volumes of textual content. In Oracle Database 12c this index has been enhanced so that it fully understands the XML data model used by XQuery Full text. The index can now support XQuery Full text based searches in the same way that it supports CONTAINS based searches.

Full details of the XQuery and XPath Full Text facility 1.0 recommendation can be found at

http://www.w3.org/TR/xpath-full-text-10/.

Integration with Oracle Multitenant Architecture

The Oracle Multitenant option allows multiple Oracle databases to be consolidated into a single database instance. This makes it much easier for an organization to manage large numbers of Oracle databases. In a multitenant environment, a multitenant container database (CDB) is responsible for managing one or more pluggable databases (PDB).

Oracle XML DB has been fully integrated into the multitenant architecture. The CDB and PDB each have a private instance of the Oracle XML DB repository. Configuration of XDB services at the PDB level is still managed by the PDB's xdbconfig.xml file. The Oracle XML DB protocol servers support multitenant configurations where more than one PDB is offering HTTP and FTP services.

Due to the nature of the HTTP and FTP protocols, a unique port must be allocated for each PDB that is offering HTTP or FTP services. Since shared servers are a property of the CDB rather than the PDB, the number of shared servers required in a multitenant configuration must be sufficient to meet the combined requirements of all of the PDBs serviced by the instance. When plugging a new PDB into a multitenant configuration, new values may need to be assigned for the HTTP and FTP port numbers before the database can be opened.

When enabling anonymous access to the database, the ANONYMOUS account must first be unlocked at the CDB level. Once ANONYMOUS has been unlocked in the CDB, it can be locked or unlocked as required in each PDB. The ALLOW_ANONYMOUS_REPOSITORY_ACCESS element in the PDB's xdbconfig.xml file still controls whether or not ANONYMOUS can access public content in the PDB's Oracle XML DB repository.

HTTPS Protocol Configuration and EM Express

In Oracle Database 12c the Database Console application has been replaced by EM Express. EM Express is served directly from the database using the Oracle XML DB protocol. In order to ensure that EM Express is available as soon as a database has been opened the HTTPS protocol in now enabled by default. The default port for the HTTPS protocol is 5500, and EM Express is accessed using a servlet configured as '/em'. In a multitenant configuration, EM Express is enabled in the CDB by default.

Anytime you access the repository using the HTTPS protocol, the browser will report that there is an error with the security certificate. The browser will indicate the following problems

The security certificate presented by this website was not issued by a trusted certificate authority.
The security certificate presented by this website was issued for a different website's address.

These error messages can be safely ignored. If you want to get rid of these errors you will need to install your own security certificate in place of the default one shipped with the Oracle Database.

Digest Authentication

In Oracle Database 12c the Oracle XML DB HTTP server supports digest-based authentication. This ensures that HTTP passwords are now securely encrypted while on the wire. Note that digest authentication is only supported with HTTP. Care must still be taken when using the FTP protocol, to ensure that passwords cannot be compromised. One side effect of enabling digest authentication is that the username for HTTP authentication is now case sensitive. This means that in most

configurations the database username must be supplied in UPPERCASE when using digest authentication with HTTP.

XMLTable Optimizations for Binary XML

Significant work has been done to improve the performance of XMLTable operations on top of the binary XML storage model. This work involved creating a new 'row source' which allows XMLTable operations on binary XML to operate at least two times faster in Oracle Database 12c as compared to Oracle Database 11g (11.2.0.3.0). As well as speeding up XMLTable operations, these optimizations also improve the performance for index maintenance operations associated with structured XML Index.

Partitioning and Parallel Processing Improvements

Many of the XML features provided by Oracle XML DB have been enhanced to take full advantage of the parallelism inherent in Oracle Database 12c. Examples of this include

- Adding parallel index maintenance for Structured XML Index.
- Supporting Range, List and Hash partitioning with Structured and Unstructured XML Index.
- Enabling Parallel Query operations for XQuery and SQL/XML.

Providing significantly enhanced support for parallel query and indexing allows Oracle XML DB to take full advantage of the scalability offered by today's massively parallel platforms such as Oracle's latest generation of EXADATA managed systems.

Replication and Availability

Oracle Database 12c enables all kinds of XMLType persistence to be used with Oracle's popular Logical Standby option. In addition replication of all forms of XMLType storage via Oracle Golden Gate is now supported.

Rolling Upgrade

An Oracle database with Oracle XML DB installed can now participate in a rolling upgrade. This makes it much easier to schedule database upgrades in a RAC configuration by eliminating the needs for a database service outage.

XML Manageability Packages

Oracle Database 12c is the first release of the Oracle Database to have the XML manageability packages installed by default. These packages make it much easier to do common XML DB related tasks. There are currently two XML manageability packages:

- DBMS_XMLSCHEMA_ANNOTATE: This package enables programmatic annotation of an XML Schema prior to registering it with the database. This makes it easier to use a consistent set of annotations when working with different versions of an XML Schema.

- DBMS_XMLSTORAGE_MANAGE: This package simplifies common tasks related to managing XML storage, including renaming collection tables and creating B-Tree indexes on object-relational storage.

Conclusion

Organizations are facing a growing volume of XML content that they need to deal with. Organizations need to adopt a platform that can manage XML with a similar degree of rigor and security to the one used for their relational data. When managing data, organizations need to avoid stove-pipe solutions, where a separate product is used for each kind of content they need to manage. Stove-pipe solutions make it difficult to share data, and incur significant additional costs arising from the acquisition and operation of each solution. Organizations also need to avoid a lowest common denominator approach, where the valuable information contained in their XML get locked up in a key-value or document store which cannot properly query, index and manipulate the content it is managing.

Oracle XML DB provides all of the features and functionality and performance required to manage an organization's XML content. Oracle XML DB, and other technologies such as Oracle Text and Oracle Spatial and Graph, allow organizations to manage all of their mission critical data using a single scalable, available and reliable platform, namely Oracle Database 12c.

Oracle XML DB in Oracle Database 12c
June 2013
Author: Mark D Drake
Contributing Authors:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together