

Oracle Database 10g Release 2 XML DB Technical Overview

An Oracle White Paper
May 2005

Oracle Database 10g Release 2 XML DB Technical Overview

Introduction.....	1
Oracle XML DB Architecture	2
XMLType Storage	2
Oracle XML DB Repository	4
Accessing and Manipulating XML in the Oracle XML DB Repository	4
XML Services	4
Repository Metadata	5
Oracle XML DB Repository Architecture	6
How Does Oracle XML DB Repository Work?	6
Oracle XML DB Protocol Architecture.....	8
APIs for Oracle XML DB (Java, PL/SQL, ODP.NET, and C)	8
Oracle XML DB Capabilities	8
XMLType	9
Oracle XML DB Stores XML Text in CLOBs.....	9
XMLType Tables and Columns Can Conform to an XML Schema	10
The XMLType API.....	10
XML Schema	10
XML Schema Unifies Document and Data Modeling.....	11
You Can Create XMLType Tables and Columns, Ensure DOM Fidelity	11
Use XMLType Views to Wrap Relational Data	11
W3C's Schema for Schemas	11
XML Schema's Base Set of Data Types Can be Extended.....	12
Unstructured Versus Structured Storage.....	12
Unstructured Storage of XML Documents.....	12
Structured Storage of XML Documents	12
Using Indexes to Improve Performance of XPath-Based Functions.....	13
Guidelines for Choosing the Storage Options	13
SQL/XML Duality	15
SQL Standard SQL/XML Functions	16
XPath Rewrite.....	17
Oracle Database-Native XQuery	19
FLWOR Expressions in XQuery	20

XMLQuery SQL Function	20
XMLTable SQL Construct	20
Rewrite for XQuery	21
Searching XML Data Stored in CLOBs Using Oracle Text.....	21
Managing Oracle XML DB Applications with Oracle Enterprise Manager	22
Oracle XML DB Benefits	23
Unifying Data and Content	23
Exploiting Database Capabilities.....	24
Exploiting XML Capabilities.....	25
Faster Storage and Retrieval of Complex XML Documents	26
Helps You Integrate Applications.....	26
XMLType Views of Non-XML Data	26
Conclusion	27

Oracle Database 10g Release 2 XML DB Technical Overview

The vision of Oracle XML DB is to unify the two traditionally separate worlds of data and content management by building on this primary strength of XML.

INTRODUCTION

XML has been propelling the transformation of the World Wide Web into the next generation. While the first generation of the World Wide Web has made revolutionary changes across many industries, the disruptive impact of its next generation may be even more profound. One of the key strengths of XML is its ability to represent both structured data and unstructured data. The vision of Oracle XML DB is to unify the two traditionally separate worlds of data and content management by building on this primary strength of XML.

Since its introduction in Oracle9i Release 2, XML DB has received phenomenal reviews for doing the best job in hiding the complexity of managing XML data, offering the richest set of query capabilities, and delivering extras such as schema evolution and WebDAV. As a result, the adoption of XML DB has intensified among enterprises to become mainstream in unified data and content management.

Built on the solid foundation of Oracle database, Oracle XML DB provides extensive capabilities for the efficient storage, retrieval, querying, generation, and management of massive volumes of XML data. Since its inception, Oracle XML DB has provided deep integration of XMLType, XML Schema processing, structured and unstructured storage, content repository, SQL/XML, and management capabilities. With the release of each new Oracle database, XML DB has also continued to evolve with more versatile, scalable, and efficient features in many key areas. In the new Oracle Database 10g Release 2, Oracle XML DB introduces an efficient implementation of standards-based XQuery capabilities, a versatile schema-based resource metadata facility, a set of new SQL functions for DML operations on XML data, and much more.

In this paper, we will provide a comprehensive examination of Oracle XML DB and its foundation. We will begin by describing the architecture of XML DB before examining its major capabilities.

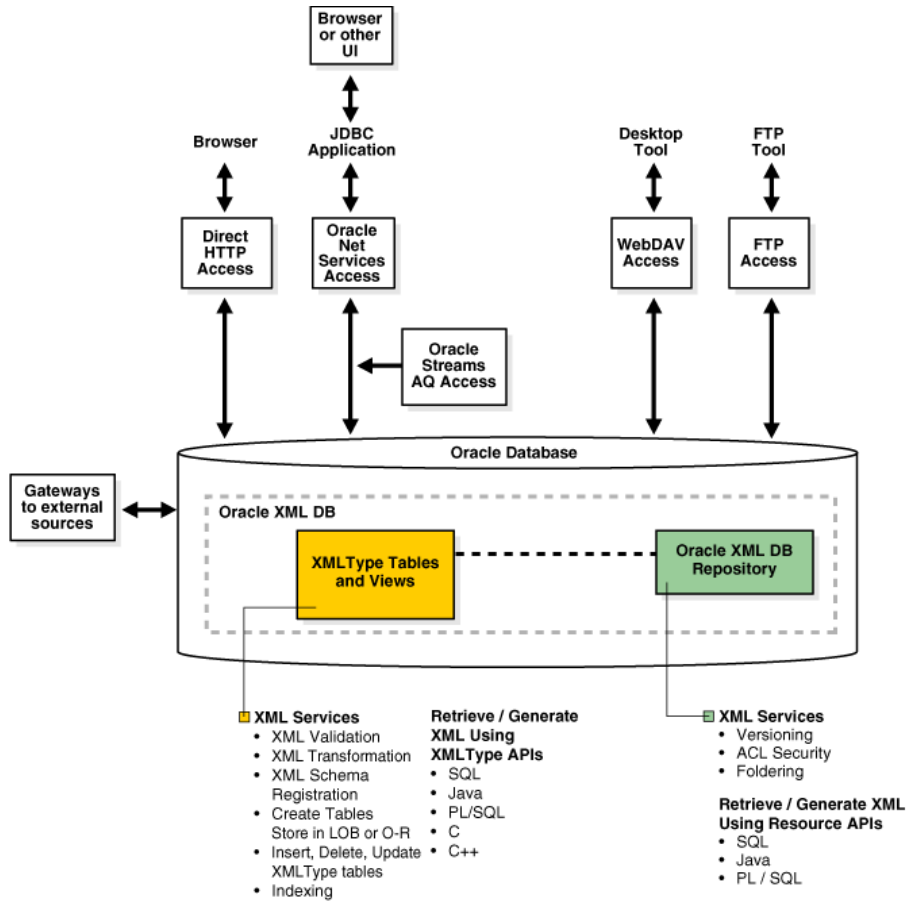
ORACLE XML DB ARCHITECTURE

As shown in Figure 1 below, Oracle XML DB has two primary pillars:

- Storage of XMLType tables and views
- Oracle XML DB Repository

There are a rich set of associated services, protocols, and APIs for these two pillars to support numerous usage scenarios.

Figure 1 XMLType Storage and Repository



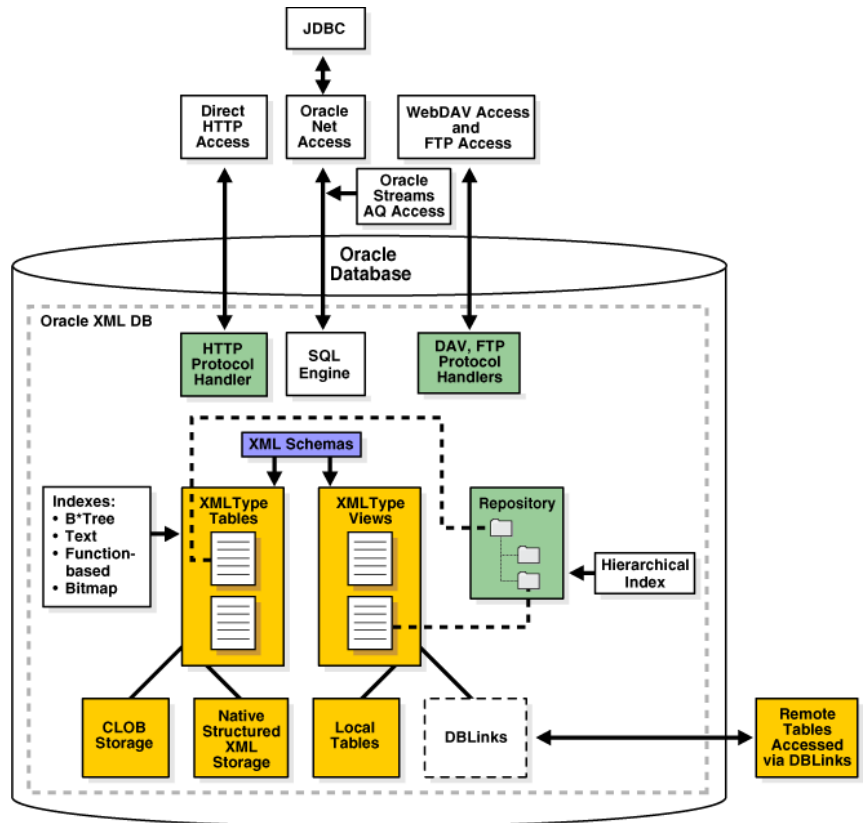
XMLType Storage

As shown in Figure 2 below, XMLType storage in Oracle XML DB has a number of essential components. When XML schemas are registered with Oracle XML DB, a set of default tables are created and used to store XML instance documents associated with the schema. These documents can also be viewed and accessed in Oracle XML DB Repository.

XMLType tables and columns can be stored as Character Large Object (CLOB) values or as a set of objects. When stored as a set of objects, we refer to structured, or shredded storage.

Data in XMLType views can be stored in local or remote tables. Remote tables can be accessed through database links.

Figure 2 XMLType Storage



Both XMLType tables and views can be indexed using B*Tree, Oracle Text, function-based indexes, or bitmap indexes.

You can access data in Oracle XML DB Repository by using any of the following:

- Concurrent HTTP and HTTPS, through the HTTP protocol handler.
- WebDAV and FTP, through the WebDAV and FTP protocol server.
- SQL, through Oracle Net Services, including JDBC.

Oracle XML DB supports XML data messaging using Oracle Streams Advanced Queuing (AQ) and Web Services.

Oracle XML DB Repository

Oracle XML DB Repository is a component of Oracle Database that is optimized for handling XML data. The Oracle XML DB repository contains resources. Resources can be either folders (directories, containers) or files. Each resource is identified by a path. A resource also has a set of system-defined metadata (such as `Owner` and `CreationDate`), in addition to its content. It may also have user-defined metadata – that is, information that is not part of the content, but is associated with it.

Although Oracle XML DB Repository handles XML content in particular, you can use Oracle XML DB Repository to store other kinds of data, besides XML; in fact, you can use the repository to access any data that is stored in Oracle Database.

Accessing and Manipulating XML in the Oracle XML DB Repository

You can access data in Oracle XML DB Repository in the following ways:

- Using SQL, through views `RESOURCE_VIEW` and `PATH_VIEW`
- Using PL/SQL, through the `DBML_XDB` API
- Using Java, through the Oracle XML DB resource API for Java

XML Services

Besides supporting APIs that access and manipulate data, Oracle XML DB Repository provides APIs for the following services:

- **Versioning.** Oracle XML DB uses the `DBMS_XDB_VERSION` PL/SQL package for versioning resources in Oracle XML DB Repository. Subsequent updates to a resource create a new version (the data corresponding to previous versions is retained). Versioning support is based on the IETF WebDAV standard.
- **ACL Security.** Oracle XML DB resource security is based on Access Control Lists (ACLs). Every resource in Oracle XML DB has an associated ACL that lists its privileges. Whenever resources are accessed or manipulated, the ACLs determine if the operation is legal. An ACL is an XML document that contains a set of Access Control Entries (ACE). Each ACE grants or revokes a set of permissions to a particular user or group (database role). This access control mechanism is based on the WebDAV specification.
- **Foldering.** Oracle XML DB Repository manages a persistent hierarchy of folder (directory) resources that contain other resources. Oracle XML DB modules, such as protocol servers, the schema manager, and the Oracle XML DB `RESOURCE_VIEW` API, use foldering to map path names to resources.

Repository Metadata

Data that you use is often associated with additional information that is not part of the content. You can use such metadata to group or classify data and to process it in different ways. For example, you might have a collection of digital photographs and you might associate metadata with each picture, such as information about the photographic characteristics (color composition, focal length) or context (location, kind of subject: landscape, people).

An Oracle XML DB repository resource is an XML document that contains both metadata and data; the data is the contents of element `Contents`; all other elements in the resource contain metadata. The data of a resource can of course be XML or not.

You can associate resources in the Oracle XML DB repository with metadata that you define. In addition to such **user-defined metadata**, each repository resource also has associated metadata that Oracle XML DB creates automatically and uses (transparently) to manage the resource. Such **system-defined metadata** includes properties such as the owner and creation date of each resource.

Except for system-defined metadata, you decide what resource information should be treated as data and what should be treated as metadata. In the case of a photo resource, supplemental information about the photo is normally not considered to be part of the photo data, which is a binary image. For text, however, you sometimes have a choice of whether to include particular information in the resource contents (data) or keep it separate and associate it with the contents as metadata — that choice is often influenced by the applications that use or produce the data.

User-Defined Resource Metadata

User-defined resource metadata is itself represented as XML: it is XML data that is associated with other XML data, describing it or providing it with supplementary, related information.

User-defined metadata for resources can be either XML schema-based or not:

- Schema-based resource metadata is new in Oracle Database 10g Release 2. It is stored in separate (out-of-line) tables. These are related to the resource table by the resource OID, which is stored in the hidden object column `RESID` of the metadata tables.
- Resource metadata that is not schema-based is stored in a `CLOB` column (`RESEXTRA`) in the resource table.

You can take advantage of schema-based metadata, in particular, to perform efficient queries and DML operations on resources. You can perform the following tasks involving schema-based resource metadata:

- Create and register an XML schema that defines the metadata for a particular kind of resource.
- Add metadata to a repository resource and update (modify) such metadata.
- Query resource metadata to find associated content.
- Delete specific metadata associated with a resource and purge all metadata associated with a resource.

In addition, you can also add non-schema-based metadata to a resource.

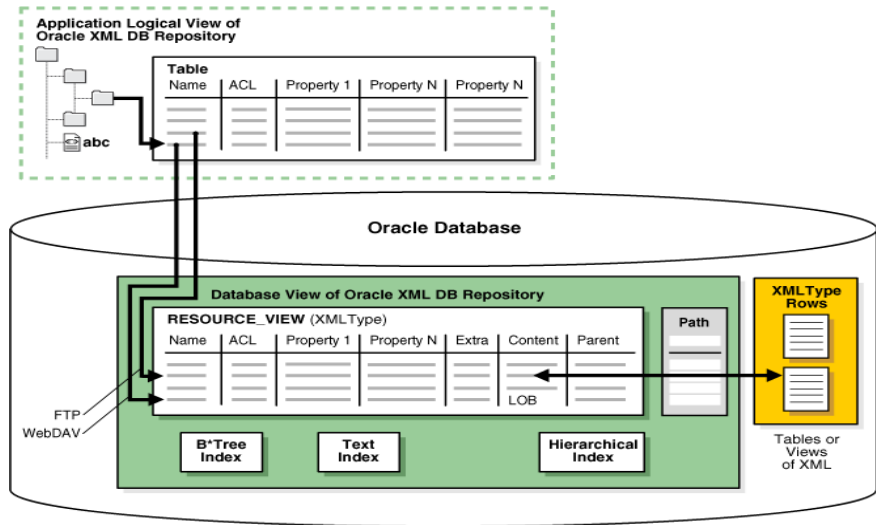
Typical uses of resource metadata include workflow applications, user rights management, tracking resource ownership, and controlling resource validity dates.

Oracle XML DB Repository Architecture

In figure 3, we depict the Oracle XML DB Repository architecture. A resource is any piece of content managed by Oracle XML DB. Each resource has a name, an associated access control list that determines who can see the resource, certain static properties, and additional properties that are extensible by the application. You can access the repository in SQL, for example, by using the `RESOURCE_VIEW` API.

In addition to the resource information, the `RESOURCE_VIEW` also contains a `Path` column, which holds the paths to each resource.

Figure 3 Oracle XML DB Repository Architecture



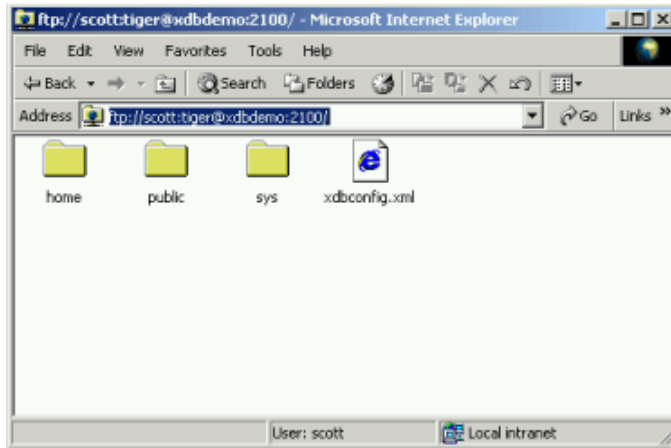
How Does Oracle XML DB Repository Work?

The relational model table-row-column metaphor, is accepted as an effective mechanism for managing structured data. The model is not as effective for managing semistructured and unstructured data, such as document- or content-

oriented XML. For example, a book is not easily represented as a set of rows in a table. It is more natural to represent a book as a hierarchy, book:chapter:section:paragraph, and to represent the hierarchy as a set of folders and subfolders.

- A hierarchical metaphor manages document-centric XML content. Relational databases are traditionally poor at managing hierarchical structures and traversing a path or URL. Oracle XML DB provides a hierarchically organized repository that can be queried and through which document-centric XML content can be managed.
- A hierarchical index speeds up folder and path traversals. Oracle XML DB includes a new, patented hierarchical index that speeds up folder and path traversals in Oracle XML DB Repository. The hierarchical index is transparent to end users, and allows Oracle XML DB to perform folder and path traversals at speeds comparable to or faster than conventional file systems.
- You can access XML documents in Oracle XML DB Repository using standard connect-access protocols such as FTP, HTTP, HTTPS, and WebDAV, in addition to languages SQL, PL/SQL, Java, and C. The repository provides content authors and editors direct access to XML content stored in Oracle Database.
- A resource in this context is a file or folder, identified by a URL. WebDAV is an IETF standard that defines a set of extensions to the HTTP protocol. It allows an HTTP server to act as a file server for a DAV-enabled client. The WebDAV standard uses the term resource to describe a file or a folder. Every resource managed by a WebDAV server is identified by a URL. Oracle XML DB adds native support to Oracle Database for these protocols. The protocols were designed for document-centric operations. By providing support for these protocols, Oracle XML DB allows Windows Explorer, Microsoft Office, and products from vendors such as Altova, Macromedia, and Adobe, to work directly with XML content stored in Oracle XML DB Repository. Figure 4 shows the root-level directory of the repository as seen from Microsoft Web Folder.

Figure 4 Microsoft Web Folder View of Oracle XML DB Repository



Hence, WebDAV clients such as Microsoft Windows Explorer can connect directly to Oracle XML DB Repository. No additional Oracle Database or Microsoft-specific software or other complex middleware is needed. End users can work directly with Oracle XML DB Repository using familiar tools and interfaces.

Oracle XML DB Protocol Architecture

One key feature of the Oracle XML DB architecture is that HTTP, HTTPS, WebDAV, and FTP protocols are supported using the same architecture used to support Oracle Data Provider for .NET (ODP.NET) in a shared server configuration. The Listener listens for HTTP, HTTPS, and FTP requests in the same way that it listens for ODP .NET service requests. When the Listener receives an HTTP, HTTPS, or FTP request, it hands it off to an Oracle Database shared server process which services it and sends the appropriate response back to the client. You can use the TNS Listener command `lsnrctl status` to verify that HTTP, HTTPS, and FTP support has been enabled.

APIs for Oracle XML DB (Java, PL/SQL, ODP.NET, and C)

All Oracle XML DB functionality is accessible from JDBC, PL/SQL, OCI, and ODP.NET APIs. The most popular methods for building web-based applications are based on the J2EE or the .NET architecture. With the J2EE architecture, JDBC can be used to access XML DB. In the .NET environment, Oracle's ODP.NET API allows access to Oracle XML DB using Visual Basic, C#, or C/C++ languages.

ORACLE XML DB CAPABILITIES

Any database used for managing XML must be able to store XML documents. Oracle XML DB is capable of much more than this. It provides standard database features such as transaction control, data integrity, replication, reliability, availability, security, and scalability, while also allowing for efficient

indexing, querying, updating, and searching of XML documents in an XML-centric manner.

The hierarchical nature of XML presents the traditional relational database with a number of challenges:

- In a relational database the table-row metaphor locates content. Primary-Key Foreign-Key relationships help define the relationships between content. Content is accessed and updated using the table-row-column metaphor.
- XML on the other hand uses hierarchical techniques to achieve the same functionality. A URL is used to locate an XML document. URL-based standards such as XLink are used to define the relationships between XML documents. W3C Recommendations like XPath are used to access and update content contained within XML documents. Both URLs and XPath expressions are based on hierarchical metaphors. A URL uses a path through a folder hierarchy to identify a document whereas XPath uses a path through an XML document's node hierarchy to access part of an XML document.

Oracle XML DB addresses these challenges by introducing new SQL functions and methods that allow the use of XML-centric metaphors, such as XPath expressions for querying and updating XML Documents.

XMLType

`XMLType` is a native server datatype that allows the database to understand that a column or table contains XML. This is similar to the way that the `DATE` datatype allows the database to understand that a column contains a date. `XMLType` also provides methods that allow common operations such as XML Schema validation and XSL transformations on XML content. You can use the `XMLType` data-type like any other datatype. For example, you can use `XMLType` when:

- Creating a column in a relational table
- Declaring PL/SQL variables
- Defining and calling PL/SQL procedures and functions

Since `XMLType` is an object type, you can also create a table of `XMLType`. By default, an `XMLType` table or column can contain any well-formed XML document.

Oracle XML DB Stores XML Text in CLOBs

Oracle XML DB stores the content of the document as XML text using the Character Large Object (`CLOB`) datatype. This allows for maximum flexibility in terms of the shape of the XML structures that can be stored in a single table or column and the highest rates of ingestion and retrieval.

XMLType Tables and Columns Can Conform to an XML Schema

XMLType tables or columns can be constrained and conform to an XML schema. This has several advantages:

- The database will ensure that only XML documents that validate against the XML schema can be stored in the column or table.
- Since the contents of the table or column conform to a known XML structure, Oracle XML DB can use the information contained in the XML schema to provide more intelligent query and update processing of the XML.
- Constraining the XMLType to an XML schema provides the option of storing the content of the document using structured-storage techniques. Structured-storage decomposes or 'shreds' the content of the XML document and stores it as a set of SQL objects rather than simply storing the document as text in a CLOB. The object-model used to store the document is automatically derived from the contents of the XML schema.

The XMLType API

The XMLType datatype provides the following structures:

- Constructors. These allow an XMLType value to be created from a VARCHAR, CLOB, BLOB, or BFILE value.
- Methods. A number of XML-specific methods that can operate on XMLType objects. The methods provided by XMLType provide support for common operations:
 - Construct an XMLType instance – the `createXML()` method
 - Extract a subset of nodes contained in the XMLType – the `extract()` method
 - Check whether or not a particular node exists in the XMLType – the `existsNode()` method
 - Validate the contents of the XMLType against an XML schema – the `schemaValidate()` method
 - DML operations on an instance of XMLType – the `appendChildXML()`, `insertXMLBefore()`, and `deleteXML()` methods
 - Perform an XSL Transformation – the `transform()` method

XML Schema

Support for the Worldwide Web Consortium (W3C) XML Schema Recommendation is a key feature in Oracle XML DB. XML Schema specifies the structure, content, and certain semantics of a set of XML documents.

XML Schema Unifies Document and Data Modeling

XML Schema unifies both document and data modeling. In Oracle XML DB, you can create tables and types automatically using XML Schema. In short, this means that you can develop and use a standard data model for all your data, structured, unstructured, and semi-structured. You can use Oracle XML DB to enforce this data model for all your data.

You Can Create XMLType Tables and Columns, Ensure DOM Fidelity

You can create XML schema-based `XMLType` tables and columns and optionally specify, for example, that they:

- Conform to pre-registered XML schemas
- Are stored in structured storage format specified by the XML schema, maintaining DOM fidelity

Use XMLType Views to Wrap Relational Data

You can also choose to wrap existing relational and object-relational data into XML format using `XMLType` views.

You can store an `XMLType` object as an XML object that is based on an XML schema or not based on an XML schema:

- schema-based objects. These are stored in Oracle XML DB as Large Objects (LOBs) or in structured storage (object-rationally) in tables, columns, or views.
- Non-schema-based objects. These are stored in Oracle XML DB as LOBs.

You can map from XML instances to structured or LOB storage. The mapping can be specified in an XML schema, and the schema must be registered in Oracle XML DB. This is a required step before storing XML schema-based instance documents. Once registered, the XML schema can be referenced using its URL.

W3C's Schema for Schemas

The W3C Schema Working Group publishes an XML schema, often referred to as the "Schema for Schemas". This XML schema provides the definition, or vocabulary, of the XML Schema language. An XML schema definition (XSD) is an XML document, that is compliant with the vocabulary defined by the "Schema for Schemas". An XML schema uses vocabulary defined by W3C XML Schema Working Group to create a collection of type definitions and element declarations that declare a shared vocabulary for describing the contents and structure of a new class of XML documents.

XML Schema's Base Set of Data Types Can be Extended

The XML Schema language provides strong typing of elements and attributes. It defines 47 scalar data types. The base set of data types can be extended using object-oriented techniques like inheritance and extension to define more complex types. W3C XML Schema vocabulary also includes constructs that allow the definition of complex types, substitution groups, repeating sets, nesting, ordering, and so on. Oracle XML DB supports all of constructs defined by the XML Schema Recommendation, except for redefines.

XML schema are most commonly used as a mechanism for validating that instance documents conform with their specifications. Oracle XML DB includes methods and SQL functions that allow an XML schema to be used for this.

Unstructured Versus Structured Storage

One key decision to make when using Oracle XML DB for storing XML documents is whether to use the unstructured or the structured storage approach. To improve query performance, you also need to choose the right indexing scheme(s) for both of these storage approaches.

Unstructured Storage of XML Documents

With Unstructured storage, an entire XML document is stored as a whole in a Character Large Object (CLOB). Unstructured storage provides the highest possible throughput when inserting and retrieving entire XML documents. Unstructured storage maintains document fidelity by preserving the original XML document, including white spaces. It also provides the greatest degree of flexibility in terms of the structure of the XML that can be stored in an `XMLType` table or column. These throughput and flexibility benefits come at the expense of certain aspects of intelligent processing. Little can be done by the database to optimize queries or updates on XML documents stored using a CLOB data type.

Unstructured storage provides the highest possible throughput when inserting and retrieving entire XML documents.

Structured Storage of XML Documents

Structured storage has numerous advantages in managing XML documents, including optimized memory management, reduced storage requirements, B-tree indexing, and in-place updates. For complex-structured XML documents, structured storage provides a number of options for the optimal storage of collections according to actual usage scenarios. Structured storage does require somewhat increased processing of the corresponding XML schema during ingestion and retrieval of entire documents.

For complex-structured XML documents, structured storage provides a number of options for optimal storage of collections according to actual usage scenarios

Structured storage of XML documents is based on decomposing the content of the document into a set of SQL objects. These SQL objects are based on the SQL 1999 standard. When an XML schema is registered with Oracle XML DB, the required SQL type definitions are automatically generated from the XML schema.

A SQL type definition is generated from each `complexType` defined by the XML schema. Each element or attribute defined by the `complexType` becomes a SQL attribute in the corresponding SQL type. Oracle XML DB automatically maps the 47 scalar data types defined by the W3C XML Schema Recommendation to the 19 scalar datatypes supported by SQL.

The generated SQL types allow XML content, compliant with the XML schema, to be decomposed and stored in the database as a set of objects without any loss of information. When the document is ingested, the constructs defined by the XML schema are mapped directly to the equivalent SQL types. This allows Oracle XML DB to leverage the full power of Oracle Database when managing XML and can lead to significant reductions in the amount of space required to store the document. It can also reduce the amount of memory required to query and update XML content.

Using Indexes to Improve Performance of XPath-Based Functions

For structured storage, Oracle XML DB supports the creation of three kinds of index on XML content:

- **B-Tree indexes.** When the `XMLType` table or column is based on structured storage techniques, conventional B-Tree indexes can be created on underlying SQL types.
- **Function-based indexes.** These can be created on any `XMLType` table or column. Its usage is limited to pre-defined XPath queries on unstructured storage.
- **Text-based indexes.** These can be created on any `XMLType` table or column.

Indexes are typically created by using SQL function `extractValue`, although it is also possible to create indexes based on other functions such as `existsNode`. During the index creation process Oracle XML DB uses XPath rewrite to determine whether it is possible to map between the nodes referenced in the XPath expression used in the `CREATE INDEX` statement and the attributes of the underlying SQL types. If the nodes in the XPath expression can be mapped to attributes of the SQL types, then the index is created as a conventional B-Tree index on the underlying SQL objects. If the XPath expression cannot be restated using object-relational SQL then a function-based index is created.

Guidelines for Choosing the Storage Options

To help you choose a storage option for your specific usage scenarios, the table below outlines the merits of structured and unstructured storage.

	Unstructured Storage	Structured Storage
--	-----------------------------	---------------------------

	Unstructured Storage	Structured Storage
Throughput	Highest possible throughput when ingesting and retrieving the entire content of an XML document.	The decomposition process results in slightly reduced throughput when ingesting retrieving the entire content of an XML document.
Flexibility	Provides the maximum amount of flexibility in terms of the structure of the XML documents that can be stored in an XMLType column or table.	Limited Flexibility. Only documents that conform to the XML schema can be stored in the XMLType table or column.
XML Fidelity	Delivers Document Fidelity: Maintains the original XML byte for byte, which may be important to some applications.	DOM Fidelity: A DOM created from an XML document that has been stored in the database will be identical to a DOM created from the original document. However trailing new lines, white space characters between tags and some data formatting may be lost.
Update Operations	When any part of the document is updated the entire document must be written back to disk.	The majority of update operations can be performed using XPath rewrite. This allows in-place, piece-wise update, leading to significantly reduced response times and greater throughput.
XPath-based queries	XPath operations can take advantage of function-based index on XMLType columns. Without an index, functional evaluation of XPath operations can be very expensive.	XPath operations may be evaluated using XPath rewrite, leading to significantly improved performance, particularly with large collections of documents.
SQL Constraint	SQL constraints are not currently available.	SQL constraints are supported.
Indexing	Text and function-based	B-Tree, text, and function-based

	Unstructured Storage	Structured Storage
Support	indexes.	indexes.
Optimized Memory Management	Other than XPath operations, XML operations on the document require creating a DOM from the document.	XML operations can be optimized to reduce memory requirements.

Much valuable information in an organization is in the form of semi-structured and unstructured data. Typically this data is in files stored on a file server or in a CLOB column inside a database. The information in these files is in proprietary- or application-specific formats. It can only be accessed through specialist tools, such as word processors or spreadsheets, or programmatically using complex, proprietary APIs. Searching across this information is limited to facilities provided by a crawler or full-text indexing.

Major reasons for the rapid adoption of XML are that it allows for:

- Stronger data management
- More open access to semi-structured and unstructured content.

Replacing proprietary file formats with XML allows organizations to achieve much higher levels of reuse of their semi-structured and unstructured data. The content can be accurately described using XML schemas. The content can be easily accessed and updated using standard APIs based on DOM and XPath.

For example, information contained in an Excel spreadsheet is only accessible to the Excel program, or to a program that uses Microsoft's COM APIs. The same information, stored in an XML document is accessible to any tool that can leverage the XML programming model. Structured data on the other hand does not suffer from these limitations. Structured data is typically stored as rows in tables within a relational database. These tables are accessed and searched using the relational model and the power and openness of SQL from a variety of tools and processing engines.

SQL/XML Duality

A key objective of Oracle XML DB is to provide XML/ SQL duality. This means that the XML programmer can leverage the power of the relational model when working with XML content, while the SQL programmer can leverage the flexibility of XML when working with relational content. This provides application developers with maximum flexibility, allowing them to use the most appropriate tools for a particular business problem.

Relational and XML Metaphors are Interchangeable: Oracle XML DB erases the traditional boundary between applications that work with structured data and

those that work with semi-structured and unstructured content. With Oracle XML DB the relational and XML metaphors become interchangeable.

XML/SQL duality means that the same data can be exposed as rows in a table and manipulated using SQL or exposed as nodes in an XML document and manipulated using techniques such as DOM or XSL transformation. Access and processing techniques are totally independent of the underlying storage format!

These features provide new, simple solutions to common business problems. For example:

- Relational data can quickly and easily be converted into HTML pages. Oracle XML DB provides new SQL functions that make it possible to generate XML directly from a SQL query. The XML can be transformed into other formats, such as HTML using the database-resident XSLT processor.
- You can easily leverage all of the information contained in their XML documents without the overhead of converting back and forth between different formats. With Oracle XML DB you can access XML content using SQL queries, On-line Analytical Processing (OLAP), and Business-Intelligence/Data Warehousing operations.
- Text, spatial data, and multimedia operations can be performed on XML Content.

SQL Standard SQL/XML Functions

New in SQL 2003 standard as Part 14, SQL/XML defines how SQL can be used in conjunction with XML in a database. Part 14 provides detailed definition of a new XML type, the values of an XML type, mappings between SQL constructs and XML constructs, and functions for generating XML from SQL data. Since Oracle Database 9i Release 2, SQL/XML features had been supported as an integral part of the XML DB. XML DB also includes a number of additional XPath-based SQL extensions to support querying, updating, and transformation of XML data. Oracle has been working closely with the SQL standard committee to standardize these extensions in the upcoming SQL 2005 standard. SQL/XML functions fall into two categories:

- Functions that make it possible to query and access XML content as part of normal SQL operations.
- Functions that provide a standard method for generating XML from the result of a SQL `SELECT` statement.

With the SQL/XML functions you can address XML content in any part of a SQL statement. They use XPath notation to traverse the XML structure and identify the node or nodes on which to operate. The ability to embed XPath expressions in SQL statements greatly simplifies XML access. The following describes briefly some of the more important SQL/XML operators:

- `existsNode`. This is used in the `WHERE` clause of a SQL statement to restrict the set of documents returned by a query. The `existsNode` SQL function takes an XPath expression and applies it to an XML document. The operator returns true (1) or false (0), depending on whether or not the document contains a node that matches the XPath expression.
- `extract`. This takes an XPath expression and returns the nodes that match the expression, as an XML document or fragment. If only a single node matches the XPath expression, then the result is a well-formed XML document. If multiple nodes match the XPath expression, then the result is a document fragment.
- `extractValue`. This takes an XPath expression and returns the corresponding leaf node. The XPath expression passed to `extractValue` should identify a single attribute or an element that has precisely one text node child. The result is returned in the appropriate SQL data type.
- `insertChildXML`. Insert XML element or attribute nodes as children of a given parent element node.
- `insertXMLbefore`. Insert XML nodes of any kind immediately before a given node (other than an attribute node).
- `appendChildXML`. Insert XML nodes of any kind as the last child nodes of a given element node.
- `deleteXML`. Delete XML nodes of any kind. The XML document that is the target of the deletion can be schema-based or non-schema-based.
- `updateXML`. This allows partial updates to be made to an XML document, based on a set of XPath expressions. Each XPath expression identifies a target node in the document, and a new value for that node. The `updateXML` operator allows multiple updates to be specified for a single XML document.
- `XMLSequence`. This makes it possible to expose the members of a collection as a virtual table

XPath Rewrite

The SQL/XML functions, and corresponding `XMLType` methods, allow XPath expressions to be used to search collections of XML documents and to access a subset of the nodes contained within an XML document. Oracle XML DB has integrated exceptional XPath rewrite technology to efficiently process XPath-based SQL/XML functions and operators.

How XPath Expressions are Evaluated by Oracle XML DB

Oracle XML DB has two methods of evaluating XPath expressions that operate on `XMLType` columns and tables. For XML:

- Stored using structured storage techniques, Oracle XML DB attempts to translate the XPath expression in a SQL/XML operator into an equivalent SQL query. The SQL query references the object-relational data structures that underpin a schema-based `XMLType`. This process is referred to as XPath rewrite. It can occur when performing queries and `UPDATE` operations.
- Stored using unstructured storage, Oracle XML DB will evaluate the XPath using functional evaluation. Functional evaluation builds a DOM tree for each XML document and then resolves the XPath programmatically using the methods provided by the DOM API. If the operation involves updating the DOM tree, the entire XML document has to be written back to disc when the operation is completed.

Efficient Processing of SQL That Contains XPath Expressions

Oracle XML DB can rewrite SQL statements that contain XPath expressions to purely relational SQL statements, which can be processed efficiently. In this way, Oracle XML DB insulates the database optimizer from having to understand XPath notation and the XML data model. The database optimizer simply processes the rewritten SQL statement in the same manner as other SQL statements.

This means that the database optimizer can derive an execution plan based on conventional relational algebra. This allows Oracle XML DB to leverage all the features of the database and ensure that SQL statements containing XPath expressions are executed in a highly scalable and efficient manner. To sum up, there is little overhead with XPath rewrites, and Oracle XML DB can execute XPath-based queries at near-relational speed, while preserving the XML abstraction.

When Can XPath Rewrite Occur?

XPath rewrite is possible when:

- A SQL statement contains SQL/XML operators or `XMLType` methods that use XPath expressions to refer to one or more nodes within a set of XML documents.
- An `XMLType` column or table containing the XML documents is associated with a registered XML schema.
- An `XMLType` column or table uses structured storage techniques to provide the underlying storage model.
- The nodes referenced by an XPath expression can be mapped, using the XML schema, to attributes of the underlying SQL object model.

What is the XPath-Rewrite Process?

XPath rewrite performs the following tasks:

1. Identify the set of XPath expressions included in the SQL statement.
2. Translate each XPath expression into an object relational SQL expression that references the tables, types, and attributes of the underlying SQL: 1999 object model.
3. Rewrite the original SQL statement into an equivalent object relational SQL statement.
4. Pass the new SQL statement to the database optimizer for plan generation and query execution.

In certain cases, XPath rewrite is not possible. This normally occurs when there is no SQL equivalent of the XPath expression. In this situation Oracle XML DB performs a functional evaluation of the XPath expressions. In general, functional evaluation of a SQL statement is more expensive than XPath rewrite, particularly if the number of documents that needs to be processed is large.

Prior to Oracle Database 10g Release 2, when documents are stored using unstructured storage (in a CLOB value), functional evaluation is necessary any time SQL functions except `existsNode` are used. Function `existsNode` will also result in functional evaluation unless a CTXXPATH index or function-based index can be used to resolve the query.

Understanding the concept of XPath rewrite, and the conditions under which it can take place, is a key step in developing Oracle XML DB applications that will deliver the required levels of scalability and performance.

Oracle Database-Native XQuery

XQuery 1.0 is an XML Query Language developed by W3C that will become the recommended query language to query XML from a variety of data sources. Various companies are adopting XQuery as the way to query XML stored in database rows or from WebServices and to construct new XML values.

On the SQL side, the XML datatype was introduced in SQL 2003 as a way to encapsulate XML in SQL. The SQL committee is now working to integrate the querying of XML using XQuery. This is being accomplished by introducing a new SQL function: *XMLQuery*, and a new construct: *XMLTable* both of which operate on XML and SQL values using XQuery. The former is known as **XQuery-centric** approach as it allows querying and constructing XML using XQuery. The latter is known as **SQL-centric** approach as it allows breaking apart the XQuery values into relational values.

Oracle Database 10g Release2 enables XQuery support in the database server through these SQL standard functions.

FLWOR Expressions in XQuery

At the heart of XQuery is the FLWOR expression that supports iteration and binding of variables to intermediate results. This kind of expression is often useful for computing joins between two or more documents and for restructuring data. The name FLWOR, pronounced "flower", reflects the keywords `for`, `let`, `where`, `order by`, and `return`.

Similar to the `FROM` clause in SQL, the `for` and `let` clauses in a FLWOR expression generate a sequence of tuples of bound variables called the **tuple stream**. Performing the same function as the `WHERE` clause in SQL, the `where` clause serves to filter the tuple stream, retaining some tuples and discarding others. The `order by` clause mimics the `ORDER BY` clause in SQL to impose an ordering on the tuple stream. Finally, the `return` clause works like the `SELECT` clause in SQL to construct the result of the FLWOR expression. The `return` clause is evaluated once for every tuple in the tuple stream, after filtering by the `where` clause, using the variable bindings in the respective tuples. The result of the FLWOR expression is an ordered sequence containing the concatenated results of these evaluations.

XMLQuery SQL Function

The XMLQuery function takes an XQuery expression as a string literal, an optional context item and other bind variables and returns the result of evaluating the XQuery expression using these input values.

Below is the syntax that will be supported in Oracle Database 10g Release 2:

```
XMLQUERY (<XQuery-string-literal>  
    [PASSING [BY VALUE] <expression-returning-XMLType>]  
    RETURNING CONTENT)
```

The XQuery string literal is a complete XQuery expression including the prolog etc. The `PASSING` clause must be followed by an expression returning an XMLType that is used as the context for evaluating the XQuery expression.

To run XQuery on XMLType columns, tables, views, or expressions generated by SQL/XML functions, it is recommended that users pass the value as an argument to the XMLQuery function. However, to query any relational table or view as XML without having to first create SQL/XML views on top of them, users can use Oracle provided XQuery function: `ora:view()` within an XQuery expression.

XMLTable SQL Construct

The XMLTable construct is used to map the result of an XQuery evaluation into relational rows and columns so that the user can query the XQuery result as a virtual relational table using SQL. The XMLTable construct can only be used in the `from` clause of SQL queries.

Below is the syntax that will be supported in Oracle Database 10g Release 2:

```
<XML table> ::=
  "XMLTable" "(" <XQuery-string-literal>
    ["PASSING" ["BY" "VALUE"] <xml-value-expression> ]
    ["COLUMNS" <XML-table-columns>]
    ")"

<XML-table-columns> ::= <XML-table-column>
    ["," <XML-table-column>]...

<XML-table-column> ::= <column-name> [<data-type>]
    [PATH <string-literal>][ "DEFAULT" <value-expression> ]
```

Rewrite for XQuery

Similar to XPath-rewrite for XPath-based SQL/XML functions, Oracle's database-native XQuery implementation excels with extensive XQuery rewrites. XQuery rewrites take full advantage of Oracle's high performance relational query engine. With XML documents stored using the structured storage approach, XQuery can be rewritten into pure relational queries to completely avoid building DOM trees of XML documents in memory. Query performance can be orders of magnitude faster with rewrites applied.

In the example below, the XQuery can be rewritten into an equivalent relational query to attain the same performance level as pure relational queries.

```
SELECT XMLQuery(
  'for $b in ora:view("SITE_TAB")/site/people/person
   where $b/@id = "person0"
   return $b/name' returning content)
FROM dual;

SELECT ( SELECT XMLAgg(XMLElement("name", p.name))
FROM SITE_TAB s, PERSON_TAB p
WHERE p.id = 'person0' AND
p.NESTED_TABLE_ID=s."SYS_NC0004700048$"
)
FROM dual;
```

Searching XML Data Stored in CLOBs Using Oracle Text

Oracle enables special indexing on XML, including Oracle Text indexes for section searching, special operators to process XML, aggregation of XML, and special optimization of queries involving XML.

XML data stored in Character Large Objects (CLOB datatype) or stored in XMLType columns in structured storage (object-rationally), can be indexed using Oracle Text. Operators `hasPath()` and `inPath()` are designed to optimize XML data searches where you can search within XML text for substring matches.

Oracle XML DB also provides:

- SQL function `CONTAINS` and XPath function `ora:contains`, which can be used with SQL function `existsNode` for XPath-based searches.

- The ability to create indexes on `URIType` and `XDBURIType` columns.
- Index type `CTXXPATH`, which allows higher performance XPath searching using `existsNode`.

Managing Oracle XML DB Applications with Oracle Enterprise Manager

You can use Oracle Enterprise Manager (Enterprise Manager) to manage and administer your Oracle XML DB application. Enterprise Manager's graphical user interface facilitates your performing the following tasks:

- Configuration
 - Configuring Oracle XML DB, including protocol server configuration
 - Viewing and editing Oracle XML DB configuration parameters
 - Registering XML schema
- Create resources
 - Managing resource security, such as editing resource ACL definitions
 - Granting and revoking resource privileges
 - Creating and editing resource indexes
 - Viewing and navigating Oracle XML DB Repository
- Create XML schema-based tables and views
 - Creating your storage infrastructure based on XML schemas
 - Editing an XML schema
 - Creating an `XMLType` table and a table with `XMLType` columns
 - Creating a view-based XML schema
 - Creating a function-based index based on XPath expressions

Figure 5 Enterprise Manager XML Schema Management

Database Instance: ora10gr2 > XML Schemas

Logged in As SYS

XML Schemas

Object Type: XML Schema

Search

Select an object type and optionally enter a schema name and an object name to filter the data that is displayed in your results set.

Schema 

Object Name

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

Selection Mode:

Select	Owner	XML Schema URL	Public
<input checked="" type="radio"/>	SCOTT	http://localhost:8080/home/SCOTT/poSource/xsd/purchaseOrder.xsd	NO

[Database](#) | [Setup](#) | [Preferences](#) | [Help](#) | [Logout](#)

Copyright © 1996, 2005, Oracle. All rights reserved.
[About Oracle Enterprise Manager 10g Database Control](#)

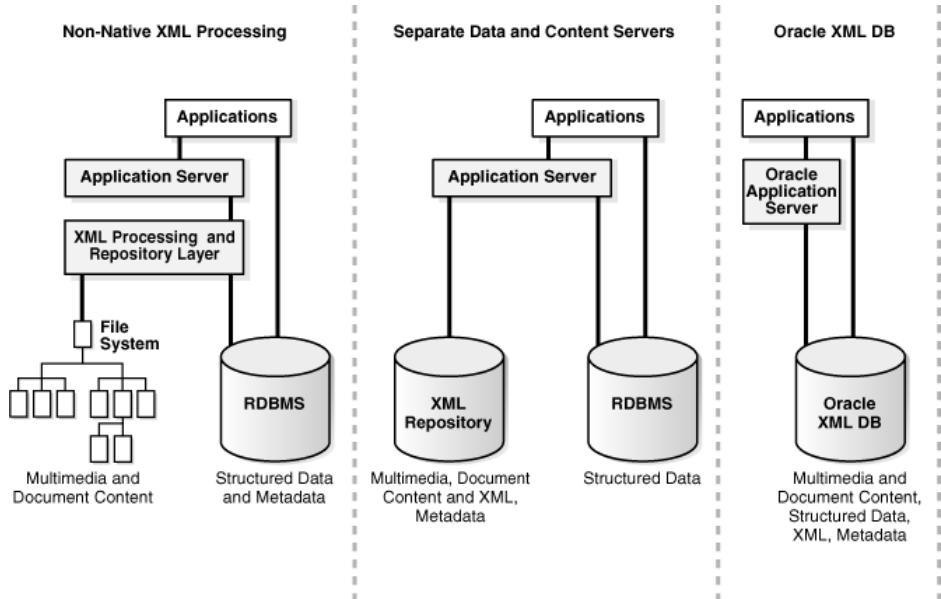
ORACLE XML DB BENEFITS

The following sections describe several benefits for using Oracle XML DB advantages.

Unifying Data and Content

Most application data and Web content is stored in a relational database or a file system, or both. XML is often used for transport, and it is generated from a database or a file system. As the volume of XML transported grows, the cost of regenerating these XML documents grows, and these storage methods become less effective at accommodating XML content.

Figure 6 Unifying Data and Content: Some Common XML Architectures



Organizations today typically manage their structured data and unstructured data differently:

- Unstructured data, in tables, makes document access transparent and table access complex
- Structured data, often in binary large objects (such as in BLOBs), makes access more complex and table access transparent.

With Oracle XML DB, you can store and manage data that is structured, unstructured, and semistructured using a standard data model and standard SQL and XML. You can perform SQL operations on XML documents and XML operations on object-relational (such as table) data.

Exploiting Database Capabilities

Oracle Database has strong XML support with the following key capabilities:

- **Indexing and Searching:** Applications use queries such as "find all the product definitions created between March and April 2002", a query that is typically supported by a B*Tree index on a date column. Oracle XML DB can enable efficient structured searches on XML data, saving content-management vendors the need to build proprietary query APIs to handle such queries.
- **Updates and Transaction Processing:** Commercial relational databases use fast updates of subparts of records, with minimal contention between users trying to update. As traditionally document-centric data participate in collaborative environments through XML, this requirement becomes more

important. File or CLOB storage cannot provide the granular concurrency control that Oracle XML DB does.

- **Managing Relationships:** Data with any structure typically has foreign key constraints. Currently, XML data-stores lack this feature, so you must implement any constraints in application code. Oracle XML DB enables you to constrain XML data according to XML schema definitions and hence achieve control over relationships that structured data has always enjoyed.
- **Multiple Views of Data:** Most enterprise applications need to group data together in different ways for different modules. This is why relational views are necessary—to allow for these multiple ways to combine data. By allowing views on XML, Oracle XML DB creates different logical abstractions on XML for, say, consumption by different types of applications.
- **Performance and Scalability:** Users expect data storage, retrieval, and query to be fast. Loading a file or CLOB value, and parsing, are typically slower than relational data access. Oracle XML DB dramatically speeds up XML storage and retrieval.
- **Ease of Development:** Databases are foremost an application platform that provides standard, easy ways to manipulate, transform, and modify individual data elements. While typical XML parsers give standard read access to XML data they do not provide an easy way to modify and store individual XML elements. Oracle XML DB supports a number of standard ways to store, modify, and retrieve data: using XML Schema, XPath, DOM, and Java.

Exploiting XML Capabilities

If the drawbacks of XML file storage force you to break down XML into database tables and columns, there are still several XML advantages you have:

- **Structure Independence:** The open content model of XML cannot be captured easily in the pure tables-and-columns world. XML schemas allow global element declarations, not just scoped to a container. Hence you can find a particular data item regardless of where in the XML document it moves to as your application evolves.
- **Storage Independence:** When you use relational design, your client programs must know where your data is stored, in what format, what table, and what the relationships are among those tables. `XMLType` enables you to write applications without that knowledge and allows DBAs to map structured data to physical table and column storage.
- **Ease of Presentation:** XML is understood natively by Web browsers, many popular desktop applications, and most Internet applications. Relational data is not generally accessible directly from applications; programming is

required to make relational data accessible to standard clients. Oracle XML DB stores data as XML and makes it available as XML outside the database; no extra programming is required to display **database content**.

- **Ease of Interchange:** XML is the language of choice in business-to-business (B2B) data exchange. If you are forced to store XML in an arbitrary table structure, you are using some kind of proprietary translation. Whenever you translate a language, information is lost and interchange suffers. By natively understanding XML and providing DOM fidelity in the storage/retrieval process, Oracle XML DB enables a clean interchange.

Faster Storage and Retrieval of Complex XML Documents

Users today face a performance barrier when storing and retrieving complex, large, or many XML documents. Oracle XML DB provides very high performance and scalability for XML operations. The major performance features are:

- Native XMLType.
- The lazily evaluated virtual DOM support.
- Database-integrated XPath and XSLT support.
- XML schema-caching support.
- CTXPath Text indexing.
- The hierarchical index over Oracle XML DB Repository.

Helps You Integrate Applications

Oracle XML DB enables data from disparate systems to be accessed through gateways and combined into one common data model. This reduces the complexity of developing applications that must deal with data from different stores.

XMLType Views of Non-XML Data

XMLType views provide a way for you to wrap existing relational and object-relational data in XML format. This is especially useful if, for example, your legacy data is not in XML but you need to migrate to an XML format. By using XMLType views, it is unnecessary to alter your application code.

To use XMLType views, you must first register an XML schema with annotations that represent the bi-directional mapping from XML to SQL object types and back to XML. An XMLType view conforming to this schema (mapping) can then be created by providing an underlying query that constructs instances of the appropriate SQL object type.

CONCLUSION

With rapidly multiplying volumes of XML data, you can no longer store and retrieve XML data in a file system or simply as LOBs in a database. Processing XML data in the mid-tier by building DOM trees has painfully met its limits. To build scalable web applications with XML, your database tier has to reduce the workload of your middle tier by processing XML data more intelligently and efficiently. Oracle XML DB meets this acute need of the information technology industry by offering a versatile array of capabilities.

Built on the solid foundation of Oracle database, Oracle XML DB provides highly extensive capabilities for the efficient storage, retrieval, querying, generation, and management of massive volume of XML data. Further deepen its integration of XMLType, XML Schema processing, structured and unstructured storage, content repository, SQL/XML, and management capabilities, XML DB continues to evolve in many key areas. In the new Oracle Database 10g Release 2, Oracle XML DB is forging ahead with an efficient implementation of standards-based XQuery capabilities, a versatile schema-based resource metadata facility, a set of new SQL functions for DML operations on XML data, and more.

In short, XML DB in Oracle Database 10g Release 2 provides the most comprehensive and efficient capabilities for developing, deploying, and managing highly scalable and versatile XML applications.

XML DB in Oracle Database 10g Release 2 provides the most comprehensive and efficient capabilities for developing, deploying, and managing highly scalable and versatile XML applications.



Oracle Database 10g Release 2 XML DB Technical Overview
May 2005
Author: Geoff Lee

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
www.oracle.com

Oracle Corporation provides the software
that powers the internet.

Oracle is a registered trademark of Oracle Corporation. Various
product and service names referenced herein may be trademarks
of Oracle Corporation. All other product and service names
mentioned may be trademarks of their respective owners.

Copyright © 2005 Oracle Corporation
All rights reserved.