Managing and Processing Office Documents in Oracle XML Database

Sabina Petride, Asha Tarachandani, Nipun Agarwal, Sam Idicula
Oracle Inc. USA
Redwood Shores CA, USA
{sabina.petride, asha.tarachandani, nipun.agarwal, sam.idicula}@oracle.com

Abstract - **Office Open XML, an XML-based file format for office data, has been standardized, adopted by Microsoft Office 2007 and supported by other major office suites like OpenOffice. The question we try to answer in this paper is where Oracle XML Database (XMLDB) stands with respect to the new advances in XML Open standards. We present the XMLDB architecture that allows integration with Office Open XML. We discuss the implications for content search, generation and validation, brought by transparently storing office content in the XMLDB Repository. We explain how to use the XML storage model, XML indexes and XMLDB Repository features for improved querability, and how to integrate Office Open XML content search with relational, non-XML data sources available in a database.**

Keywords - OOXML; Office 2010; XQuery; XMLIndex; Content Management Repository

I. INTRODUCTION

Office Open XML [1] (OOXML for now) has emerged as one of the industry standard file formats for representing word documents, spreadsheet, presentation and charts. It has been adopted by popular office applications: it is the file format for Word 2007. OpenOffice version 3.0 supports importing OOXML, with more products expected to follow.

With these emerging document standards come technical challenges. Systems are supposed to offer fast ingestion rates of data based on XML formats for data that has to be persisted on disks or filesystems, provide good querability and processing of such data, and integrate with easy to use and popular file content management applications. Moreover, users are expecting similar querability and accessibility options on their filesystem XML data as if it were stored in a database. Thus, more content management solutions have made the shift to (1) transparency with respect to the exact storage of the XML content, and (2) integration with popular document handling applications.

XMLDB has been around for almost a decade [2][3]. It allows for storing XML data in the database as a table column or in a filesystem-like Repository [4], that allows secure access to the data. Oracle XMLType is an abstraction and supports different storage models under the covers, from object relational (shredded over relational tables and views), to a native binary XML format [5].

With respect to the storage transparency requirement (1), XMLDB already offers a filesystem like abstraction of XML content stored in the database, via the XMLDB Repository. Structured as well as unstructured content can reside in the repository and accessed via WebDAV or FTP protocol, as well as via PL/SQL APIs and SQL views. Furthermore, with the Oracle SecureFiles project [6], XML content can be stored in the server or a file system with relatively little performance difference. Here, we focus mainly on (2), and on detailing how to tune the Repository storage for best performance.

With large simplifications, both OOXML and Open Document Format (ODT for now) documents are ZIP-compatible archives that contain XML files together with files describing relationships among these; most notably, the actual content of the document is stored as XML. For simplicity, for the remainder of this paper, we will be talking about OOXML, with the note that the same approach can be taken for ODT and for that matter for any archive XML-based ZIP-compatible file format.

We present the architecture of a system that allows XML content manipulated in Office 2007 or OpenOffice to be transparently handled in the XMLDB repository and illustrate the key benefits of this system:

1) By transparently storing archived XML-based files in the XMLDB Repository, XML content can be navigated in a file-system fashion (via WebDAV).

2) As the XML content internally resides in the database, we maintain all the benefits of databases over filesystems: manageability, backup and recovery, security, integration with other features of the RDBMS.

3) New data conforming to the emerging open XML standards can coexist with data stored in the database; this allows for both XML content validation based on an RDBMS, as well as for dynamic content generation.

4) Internally, the system stores OOXML content in the binary-XML format allowing for good compression and disk space management, streaming XPath evaluation, piecewise updates, improved fragment-level querability, and integration with other database features like partitioning, utilities, native binary-XML midtier processing etc.

5) XMLIndexes are built on top of the binary-XML OOXML content; since query evaluation is internally optimized for binary-XML in the presence of XMLIndexes, this model gives efficient inter-document fragment-level search and intra-document XML processing.

6) Applications need not devise or implement their own authentication and authorization policy enforcement logic; instead, one can rely on the database authentication and the Access Control List (ACL) mechanism that protects XMLDB resources.

7) Straightforward integration of OOXML content with existing Oracle applications that render query output in formats chosen by the application. For instance, integration with BI Publisher for presenting fragment-level OOXML extraction results as PDF.

The paper is organized as follows: In Section II, we give the necessary background information for understanding the Oracle XML Database, with a focus on XML storage and indexing (Section II.A) and the Repository for XML resources (Section II.B). Section III details the system architecture for storing OOXML documents in XMLDB. Special considerations on dynamic content generation and content validation make the subject of Section IV, while Section V gives an evaluation of document and fragment-level security enforcement possible for OOXML data stored in the database. In Section VI, we discuss fragment-level query processing for OOXML, and Section VII details their usage for a project tracking Oracle database application. We conclude and point to related work in Section VIII.

II. Background

We focus first on giving the necessary background information on XMLDB.

A. *XML Storage and Indexing*

XML content can be stored in the Oracle database either as large objects, in text format (CLOB), shredded as object-relational if schema-based (see [2]), or in the more recent binary-XML format (see [5]). With the binary format, XML tags are compacted into token identifiers. Besides reduced disk footprint, the binary-XML storage allows for fast query processing [5].

XML tables and columns can be indexed for improved XQuery performance. The XMLIndex [7] comes in a number of flavors: unstructured content can be fully indexed via an unstructured index, where essentially all paths in the XML content are indexed; semi-structured and structured content can be indexed via structured XML indexes, where the index creation statement specifies an XMLTABLE construct and the exact paths to be maintained by the index; finally, one can fine-tune an unstructured index by indicating that only certain types of paths be indexed via path-subsetting XML indexes. The application developer has the additional option of creating asynchronous XML indexes to defer index maintenance to a time when the server is less busy.

B. *Repository Events*

XMLDB provides an infrastructure for associating custom application code with XMLDB Repository actions. Various actions on the repository are defined as *events;* examples of events are PRE-CREATE, POST-CREATE, PRE-UPDATE, POST-UPDATE, RENDER etc. Application code, called *event handlers*, is used to integrate application logic with the XMLDB repository. For example, a recycle-bin application can be built on top of XMLDB Repository using a PRE-DELETE event handler for all folders except the recycle-bin folder that creates a hard-link to the file that is to be deleted to recycle-bin.

Application specific event handlers are loaded into the Oracle Database and associated to all or certain resources in the repository via *resource configurations*, a particular type of resource. Once associated, the event listeners are used for any repository access – SQL views, PL/SQL APIs or protocols.

III. Open Office XML Data Store and Retrieval - System Architecture

The XMLDB Repository is a filesystem-like abstraction that resides in the Oracle database and allows resources (with XML, text or binary content) to be stored and accessed either via protocols like WebDAV(RFC2518) and FTP, or via PL/SQL and JCR. Data in the repository is organized hierarchically, in folders and leaf resources, while internally it is stored in database tables.

MS Office 2007 uses WebDAV to save and open documents. The event handling mechanism of XMLDB Repository ensures that, when an OOXML document is saved under the specified path in the repository, the event handlers unzip it transparently (using the standard java.util.zip class) in the XMLDB Repository and the actual contents are moved to a Binary XML XMLType table. Similarly, all the component files are zipped on the way out of the repository at render time when the document is opened.

As the actual XML content is stored in an RDBMS, one can take advantage of the full-range of XML processing available in the database. The XML content table can be joined with relational tables present in the database. The event-based mechanism can be further exploited to dynamically build and enhance content that can be packaged to an application as OOXML, or to validate OOXML content against data available in a database. The system architecture is illustrated in Figure 1.

Figure 2 is an example of how the storage table and its index are created.

IV. Content Validation and Generation

Two main applications of this framweork are automatic content validation and generation.

A. *Content validation*

Storing the XML content of an OOXML in the database allows applications users to transparently validate the

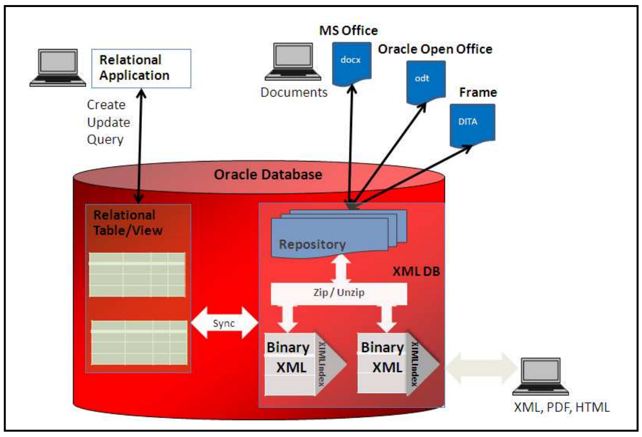


Figure 1: System architecture

OOXML content based on relevant data across multiple relational databases.

Consider for instance the case of a publisher database for managing the books submitted for review. Each book is a single Word 2007 document and, for the purpose of this section, we assume that the content is stored in a BOOK_XML table created via the statement shown in Figure 2. Author and book names, as well as the initial editing date and the actual publishing date are stored in a relational table of the form shown in Figure 3.

The system is supposed to validate the author and book names, as well as the date information in the Word document against the relational table. This can be easily incorporated into the application by issuing query checks involving the relational table BOOK_DATA and XML extract operators on the XML content in BOOK_XML. Figure 4 shows an example of a query against the XMLType table BOOK_XML that selects all the tags and their corresponding values from the document. Proper predicates with this query will achieve the desired results.

Figure 5 shows a simple query that finds the oldest publishing date of all authors who have at least one book in the category “Technical”. It involves both the XMLType table BOOK_XML and the relational table BOOK_DATA.

Using such joins, various validation rules can be applied automatically at ingestion time. For instance, to ensure that only authors of some technical books published prior to a fixed date are allowed to upload new books under a certain repository, the application event handlers can issue a query similar to Figure 5.

```
CREATE TABLE BOOK_DATA(
AUTHOR_NAME     VARCHAR2(4000),
BOOK_NAME       VARCHAR2(4000),
START_DATE      TIMESTAMP,
PUBLISHING_DATE TIMESTAMP);
```

Figure 3: Relational table

```
CREATE TABLE BOOK_XML OF XMLTYPE XMLTYPE STORE AS
BINARY XML;

CREATE INDEX BOOK_XIDX ON BOOK_XML(OBJECT_VALUE)
INDEXTYPE IS XDB.XMLINDEX
PARAMETERS('PATHS(INCLUDE(
/w:document/w:body/w:sdt//w:tbl)
NAMESPACE MAPPING
(xmlns:w="http://schemas.openxmlformats.org/wordpr
ocessingxml/2006/main"))');
```

Figure 2: Binary XML table and XMLIndex

B. *Dynamic Content Generation*

Any content that can be retrieved from the database, can be added to an OOXML document. For instance, an application that stores book-related documents may have access to various relational databases for publishing companies extra information, or book prices offered by different vendors. Such additional content may or may not be stored as XML. Applications may expect to store a book document in the repository and, upon retrieval, to get back from the repository the book document together with the corresponding data from the other databases. Another desirable usage we have encountered comes from Excel applications: as loosely formatted Excel sheets are dropped in the repository, structured parts (e.g., columns that are titled "owner", "user" or "manager") are looked up against an LDAP database and edited to include a hyperlink with "mailto: <email address retrieved from LDAP database>". This functionality is achieved by having a render event on the XML content resource issue queries on various tables,

```
SELECT BOOK_INFO.* FROM BOOK_XML BOOKS,
XMLTABLE(xmlnamespaces
('http://schemas.openxmlformats.org/wordprocessing
ml/2006/main' as "w"),
'/w:document/w:body/w:sdt'
passing BOOKS.OBJECT_VALUE
COLUMNS
TAG VARCHAR2(100) PATH
'/w:sdt/w:sdtPr/w:tag/@w:val' ,
VALUE XMLType PATH
'/w:sdt/w:sdtContent/w:p/w:r//w:t//text()' )
BOOK_INFO;

TAG         VALUE
---------------------------------------------------------
Title       The art of writing code

Category    Technical

Chapter     Chapter 1: Introduction

Chapter     Chapter 2: Understanding code

Section     Computer Languages: Similarity and
Differences

Chapter     Chapter 3: Writing code
```

Figure 4: Selecting tags and corresponding values from Word 2007

```
SELECT FRAGVAL.VAL AS "Extracted Fragment"
FROM BOOK_XML BOOKS,
     XMLTABLE( XMLNAMESPACE
      ('http://schemas.openxmlformats.org/wordprocessingml/2006/main' as "w"),
      '/w:document/w:body/w:sdt'
      PASSING BOOKS.OBJECT_VALUE
      COLUMNS TAG VARCHAR2(4000) PATH '/w:sdt/w:sdtPr/w:tag/@w:val',
              WHOLE XMLTYPE PATH '/w:sdt/w:sdtContent' ) TAGS,
     XMLTABLE( XMLNAMESPACES
      ('http://schemas.openxmlformats.org/wordprocessingml/2006/main' as "w"),
      '/w:document/w:body/w:sdt'
      PASSING BOOKS.OBJECT_VALUE
      COLUMNS TAG VARCHAR2(4000) PATH '/w:sdt/w:sdtPr/w:tag/@w:val',
              VAL XMLTYPE PATH '/w:sdt/w:sdtContent//text()') FRAGVAL
WHERE TAGS.TAG = :search_in_tag AND
      FRAGVAL.TAG=:return_tag AND
      INSTR(UPPER(TAGS.WHOLE), UPPER(:search_string))>0;
```

Figure 5: Example of OOXML join with relational data

generate XML fragments from the queries results and update the XML content with them.

V. Security and Privacy Considerations

We mentioned in Section III that a user can open a *.docx* document in Word and save it in a folder in the XMLDB repository residing in the database. The user will need to provide valid database user/password credentials in order to connect to the repository. Once the user is authenticated, access control over OOXML data residing in the repository is handled, as for any other resources in the repository, via access control list (ACL) checks.

ACL checks are by default enforced at a document level. With XMLDB integration with Office 2007, certain fragments of the documents can be tagged with ACLs and honored by the application at the fragment level.

VI. Fragment-Level Search and Retrieval

OOXML documents can be queried to retrieve information just like any other XML data. This has a large number of applications. For example:

1) Searching using XQuery across a set of documents to retrieve relevant documents or parts.

2) Extracting out a specific part of the document such as abstract, instead of whole documents, to use for re-publishing, report generation etc.

3) Extracting out information from MS Word tables embedded in documents for application uses such as aggregation, population of relational tables etc.

4) Transparently control access to search results by taking advantage of the document and fragment-level security options when storing OOXML as content in the XMLDB repository.

When certain elements are tagged using content-controls, they can be queried in the WHERE clause as well as selected out as showin in Figure 6.

Note that unlike full document search, specific parts of the document can be searched like a table or tagged elements. For example, a repository of books can be searched with queries like *"Find all books and their authors that have at least one chapter with title containing keyword*

```
SELECT BOOK.AUTHOR_NAME, min(BOOK.PUBLISHING_DATE)
FROM BOOK_DATA BOOK
WHERE BOOK.BOOK_NAME IN
 (SELECT XMLCAST(XMLQUERY('
    declare namespace
       w="http://schemas.openxmlformats.org/wordprocessingml/2006/main";
/w:document/w:body/w:sdt[w:sdtPr/w:tag/@w:val="Title"]/w:sdtContent//w:t//text()'
    PASSING BOOKS.OBJECT_VALUE
    RETURNING CONTENT) AS VARCHAR2(100) )
  FROM BOOK_XML BOOKS
  WHERE XMLEXISTS('
   declare namespace
w="http://schemas.openxmlformats.org/wordprocessingml/2006/main";
/w:document/w:body/w:sdt[w:sdtPr/w:tag/@w:val="Category"][w:sdtContent//w:t//text()="Technical"]'
   PASSING BOOKS.OBJECT_VALUE))
GROUP BY BOOK.AUTHOR_NAME
ORDER BY BOOK.AUTHOR_NAME;
```

Figure 6: Querying content control

```
SELECT doc.c1.getStringVal() “LAYER NAME”, doc.c2.getStringVal() “EFFECTS”
FROM BOOK_XML T,
    XMLTABLE( ‘XMLNAMESPACE
     ('http://schemas.openxmlformats.org/wordprocessingml/2006/main' as "w"),
     'for $i in $root//w:tbl[2]/w:tr
      where fn:contains{$i}/w:tc[1], $layer)
      return $i’
    PASSING T.OBJECT_VALUE as “root”, :p20_layer_name AS “layer”
    COLUMNS C1 XMLTYPE PATH ‘/w:tr/w:tc[1]//text()’,
            C2 XMLTYPE PATH ‘/w:tr/w:tc[2]//text()’) doc;
```

Figure 7: Query the tables in a Word document for keywords

'haunted' and have been published in the last decade” and *“Find all authors who have at least one book whose title contains keyword 'haunted' and who have had at least five publications in the last 15 years”*. All this information, even though embedded deep inside the Office documents, can be retrieved.

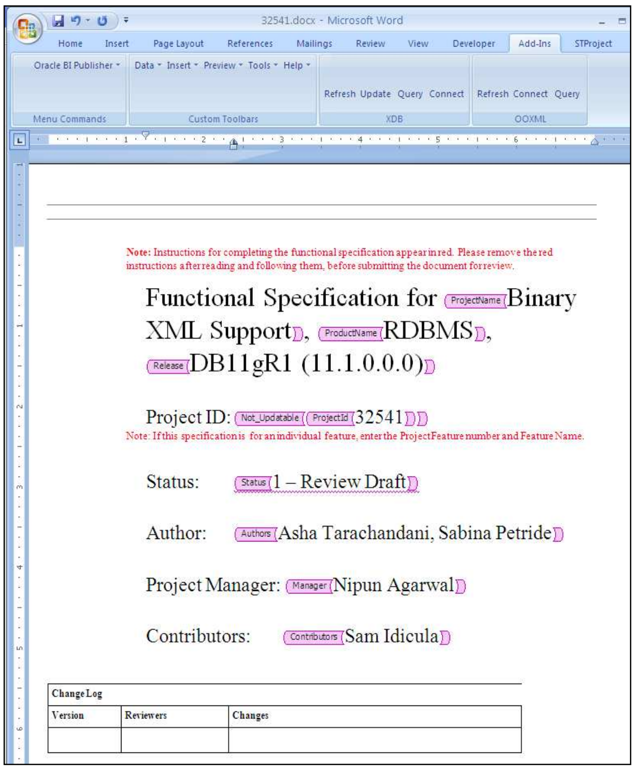
Certain parts of the documents like tables and figures, can be seearched without requiring any user input at all. For example, Figure 7 shows a query that looks for keyword defined by bind variable *:p20_layer_name* in the first column of the 2nd table of a *docx* document.

The search results can be returned in XML format and integrated with various applications. For example, **Oracle BI Publisher** can be used to display the report in various formats such as PDF, Excel sheet etc. Similarly, the search results can be utilized to generate parts of other Office documents or they can be used to populate relational tables.

VII. CASE STUDY

We have built an internal application to track development projects in various releases of Oracle database. It allows online and real-time access to product development tracking tools. Part of the process involves maintaining a database of technical specification for products. Typical technical specifications are 2MB in size on average, with about 2000 projects with technical specifications per release (up to 4TB of content per release).

This site has an estimated 19K users. The typical searches and updates are real-time, while it is not rare for a DBA to



Note: Instructions for completing the functional specification appear in red. Please remove the red instructions after reading and following them, before submitting the document for review.

Functional Specification for ProjectName Binary XML Support, ProductName RDBMS, Release DB11gR1 (11.1.0.0.0)

Project ID: Not_Updatable ProjectId 32541

Note: If this specification is for an individual feature, enter the ProjectFeature number and Feature Name.

Status: Status 1 – Review Draft

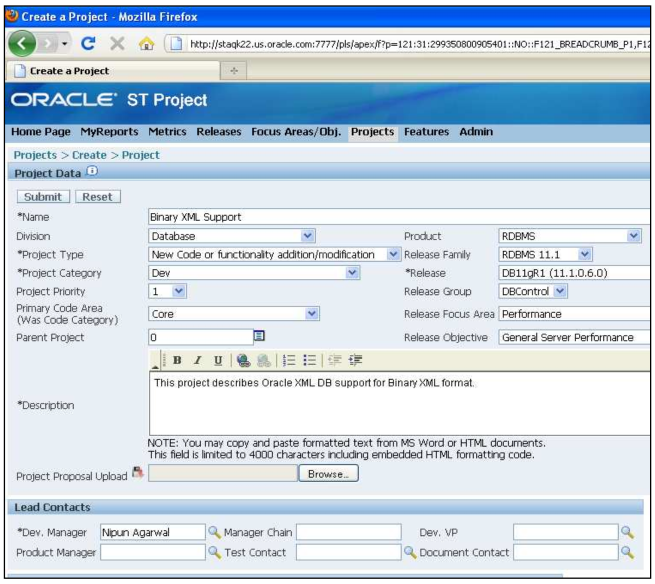
Author: Authors Asha Tarachandani, Sabina Petride

Project Manager: Manager Nipun Agarwal

Contributors: Contributors Sam Idicula

Change Log		
Version	Reviewers	Changes

Figure 8: Functional Specification Word document with custom tags



http://staqk22.us.oracle.com:7777/pls/apex/f?p=121:31:299350800905401::NO::F121_BREADCRUMB_P1,F12

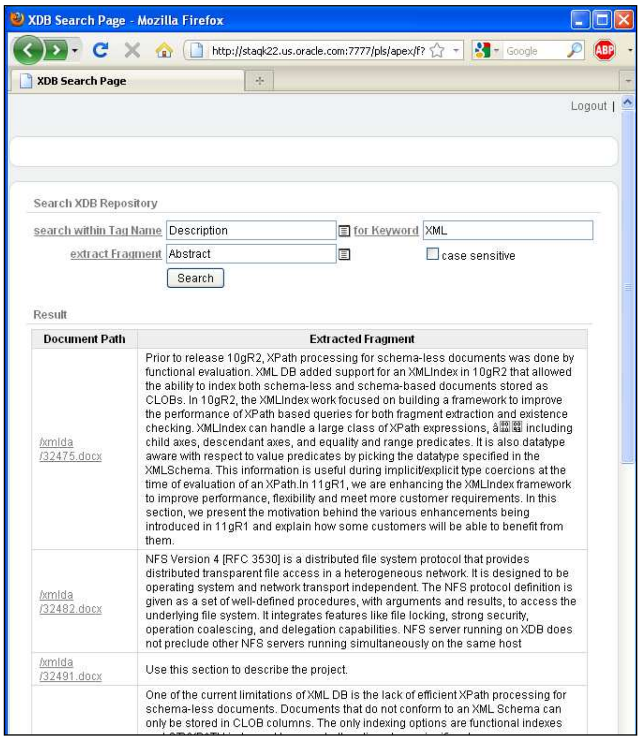
*Name	Binary XML Support		
Division	Database	Product	RDBMS
*Project Type	New Code or functionality addition/modification	Release Family	RDBMS 11.1
*Project Category	Dev	*Release	DB11gR1 (11.1.0.6.0)
Project Priority	1	Release Group	DBControl
Primary Code Area (Was Code Category)	Core	Release Focus Area	Performance
Parent Project	0	Release Objective	General Server Performance
*Description	This project describes Oracle XML DB support for Binary XML format.		

NOTE: You may copy and paste formatted text from MS Word or HTML documents.
This field is limited to 4000 characters including embedded HTML formatting code.

Project Proposal Upload

*Dev. Manager	Nipun Agarwal	Manager Chain		Dev. VP	
Product Manager		Test Contact		Document Contact	

Figure 9: Creating a project transparently generates a Word document


XDB Search Page - Mozilla Firefox
http://staqk22.us.oracle.com:7777/pls/apex/f?
Google
XDB Search Page
Logout |

Search XDB Repository

search within Tag Name Description
for Keyword XML
extract Fragment Abstract
[ ] case sensitive

Result

Document Path	Extracted Fragment
/xmlda /32475.docx	Prior to release 10gR2, XPath processing for schema-less documents was done by functional evaluation. XML DB added support for an XMLIndex in 10gR2 that allowed the ability to index both schema-less and schema-based documents stored as CLOBs. In 10gR2, the XMLIndex work focused on building a framework to improve the performance of XPath based queries for both fragment extraction and existence checking. XMLIndex can handle a large class of XPath expressions, â including child axes, descendant axes, and equality and range predicates. It is also datatype aware with respect to value predicates by picking the datatype specified in the XMLSchema. This information is useful during implicit/explicit type coercions at the time of evaluation of an XPath.In 11gR1, we are enhancing the XMLIndex framework to improve performance, flexibility and meet more customer requirements. In this section, we present the motivation behind the various enhancements being introduced in 11gR1 and explain how some customers will be able to benefit from them.
/xmlda /32482.docx	NFS Version 4 [RFC 3530] is a distributed file system protocol that provides distributed transparent file access in a heterogeneous network. It is designed to be operating system and network transport independent. The NFS protocol definition is given as a set of well-defined procedures, with arguments and results, to access the underlying file system. It integrates features like file locking, strong security, operation coalescing, and delegation capabilities. NFS server running on XDB does not preclude other NFS servers running simultaneously on the same host
/xmlda /32491.docx	Use this section to describe the project.
	One of the current limitations of XML DB is the lack of efficient XPath processing for schema-less documents. Documents that do not conform to an XML Schema can only be stored in CLOB columns. The only indexing options are functional indexes

Figure 10: Search results – look for"XML" in "Description" section and return "Abstract"

perform offline batch updates.

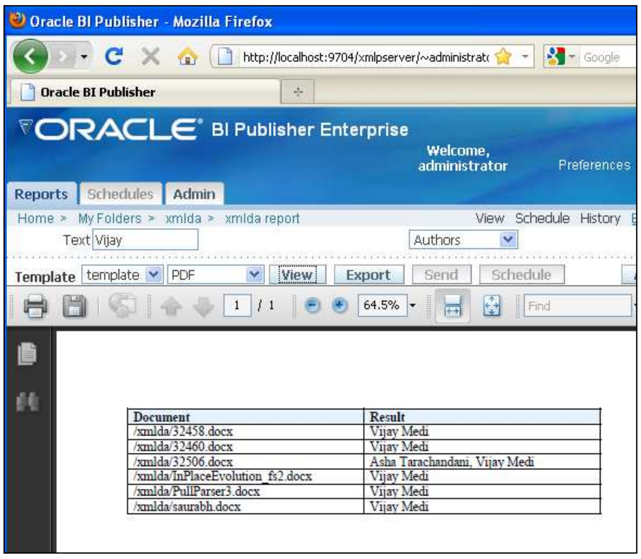
By moving the project tracking database OpenOffice content to the XMLDB repository, the existing functionality is maintained, while new functionality like fragment-based search, publishing, 2-way sync with the relational table using triggers, is gained.

Figure 8 shows a typical functional specification. Word document where fixed fields, like tile, author, project id etc. are tagged for easier search. In particular, note the *Not_Updatable* tag: the update event handlers associated with specification resources disallow changing XML content with this particular tag.

Figure 9 shows the web page for creating new projects that automatically generate *docx* file for the project. Figure 10 and Figure 11 show 2 web-based search interfaces – a standalone one and with BI publisher and result of a popular search. The search returns a document fragment matching the query, one for each specification document, as well as the Repository path of the specification, for easy full-document access.

VIII. Conclusion and Future Work

We have presented the XMLDB solution for storing, querying and rendering Open XML content. Open XML content can be saved in XMLDB as a resource in the Repository providing direct WebDAV access to Office


Oracle BI Publisher - Mozilla Firefox
http://localhost:9704/xmlpserver/~administrat
Google
Oracle BI Publisher
ORACLE BI Publisher Enterprise
Welcome, administrator
Preferences
Reports
Schedules
Admin
Home > My Folders > xmlda > xmlda report
View
Schedule
History

Text Vijay
Authors
Template template
PDF

View
Export
Send
Schedule
1 / 1
64.5%
Find

Document	Result
/xmlda/32458.docx	Vijay Medi
/xmlda/32460.docx	Vijay Medi
/xmlda/32506.docx	Asha Tarachandani, Vijay Medi
/xmlda/InPlaceEvolution_fs2.docx	Vijay Medi
/xmlda/PullParser3.docx	Vijay Medi
/xmlda/saurabh.docx	Vijay Medi

Figure 11: Search results using BI Publisher

applications. For best performance, the XML content is stored in Binary XML format with a path-subsetted XMLIndex on it. The event-based mechanism is a powerful technique allowing dynamic content generation and validation, using *any* database data. Document and fragment-level access control enforcements methods available for XMLDB resources can be also applied.

Open XML integration with content management solutions for XML is carried out successfully also by MarkLogic [8] [9]. Their toolkits for Word, Excel and Powerpoint allow Open XML data to be saved in the MarkLogic server and subsequently queried via XQuery, manipulated and rendered. The main focus of the product is on search and retrieval of text and XML granular information. It allows for search results transformations, template-based content creation, and dynamic assembly of search results. There are a number of differences between ours and their approaches.

1) XMLDB Repository being part of the Oracle database, applications storing Open XML content in XMLDB implicitly benefit from all the general database features, like high availability, backup and recovery, security, utilities etc., as well as from more recent or particular features like smart lobs and secure files we can take advantage of when choosing binary storage for Open XML content.

2) Fine tuning of the actual storage and of the indexes on top of XML content is essential for good fragment-level querability. As detailed in Section II, the application developer storing Open XML in XMLDB has the option of specifying the XML storage format and of building XML indexes tailored to a specific set of queries or applications. To the best of our knowledge, there is no equivalent of path-subsetted XML index with MarkLogic, nor is the application developer able to fine tune the storage and indexing method for particular query sets or applications.

3) XMLDB Repository events and resource configurations allow for custom and automatic work flow in applications. The application developer can use this single framework for quite different purposes, like dynamic content generation and content validation. Furthermore, this can also be used for 2-way synchronization between OOXML data and relational table with the help of event handlers and database triggers.

4) Both dynamic content generation and validation can use any data source in Oracle databases, which includes the entire XMLDB Repository. In particular, this covers non-XML, arbitrary relational data, while MarkLogic toolkits are tied to the XML content in their repositories. For the same reason, Open XML in the Oracle database is automatically available for manipulation to any database application.

Clearly, this is a functionality-only comparison. As products will become more mature and possibly other similar toolkits will be available, we expect benchmarks for Open XML and ODT handling in XML repositories to be set; we leave performance evaluations to future work.

REFERENCES

[1] "Standard ECMA-376, Office Open XML File Formats", 2006, http://www.ecma-international.org/publications/standards/Ecma-376.htm, 11.09.2010

[2] Ravi Murthy, Zhen Hua Liu, Muralidhar Krishnaprasad, Sivasankaran Chandrasekar, et. al., "Towards an enterprise XML architecture", Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 953–957, 2005.

[3] Zhen Hua Liu and Ravi Murthy, "A Decade of XML Data Management: An Industrial Experience Report from Oracle", IEEE 25th International Conference on Data Engineering, pp. 1351–1362, 2009.

[4] Ravi Murthy and Eric Sedlar, "Flexible and efficient access control in Oracle", Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pp. 973– 980, 2007.

[5] Ning Zhang, Nipun Agarwal, Sivasankaran Chandrasekar, Sam Idicula, Vijay Medi, Sabina Petride, and Balasubramanyam Sthanikam, "Binary XML Storage and Query Processing in Oracle 11g", 35th International Conference on Very Large Databases (PVLDB), volume 2, issue 2, pp. 1354– 1365, 2009.

[6] Niloy Mukherjee, Bharath Aleti, Amit Ganesh, Krishna Kunchithapadam, Scott Lynn, Sujatha Muthulingam, Kam Shergill, Shaoyu Wang and Wei Zhang, "Oracle Securefiles System". Procceedings VLDB Endowment,volume 1, issue 2, pp. 1301–1312, 2008.

[7] Geeta Arora, "XMLDB: Best Practices To Get Optimal Performance Out Of XML Queries", Oracle White Paper, June 2010, http://www.oracle.com/technetwork/database/features/xmldb/xmlqueryoptimize11gr2-168036.pdf, 11.09.2010

[8] "Dynamic Enterprise Publishing: Accelerating Information Creation, Retrieval, and Delivery with Microsoft Office and Mark Logic", MarkLogic White Paper, http://www.marklogic.com/resources/dynamic-enterprise-publishing.html, 11.09.2010

[9] Mitchell Kramer, "BlueGuru JetBlues Content Management and Publishing System", Case Study Prepared for Mark Logic by Patricia Seybold Group, 2009, http://www.scribd.com/doc/17018347/MarkLogic-at-JetBlue-Cast-Study-Blue-Guru-CMS, 11.09.2010