S311521 : Managing XML Content with Oracle XML DB and Oracle Application Express

Mark D Drake
Manager, Product Management
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Agenda

• The Oracle XML DB Repository
• APEX access to the XML DB Repository
• Oracle XML DB Repository Security
• Application principals
• Using Application principals with WebDAV
The Oracle XML DB Repository
Oracle XML DB Repository

Overview

• Content and metadata stored in the Oracle Database
• Content organized as Files in Folders rather than rows and tables
• Manages any kind of content
  – XML Specific optimizations
• Accessible via SQL and industry standard protocols
  – HTTP, FTP and WebDAV protocols
  – NFS in development
• Enables document centric development paradigm
  – Path based access to content
  – Queries based on location
Oracle XML DB Repository
Content Management Support

- Access control
  - Grant / Revoke permissions on a document by document basis

- Versioning
  - Simple linear versioning model with Check-In and Check Out

- Comprehensive event model
  - Associate code with operations on files and folders
  - Events on Files and Folders analogous to Triggers on Tables

- Standard and user defined Metadata
  - Manage metadata independently from content

- Hierarchical Index
  - Patented, high performance folder-traversal operations and queries
Oracle XML DB Repository

Standards Based

- File / Foldering model: IETF WebDAV standard
- Security: DavACL
- Protocols: HTTP, HTTPS, WebDAV, SOAP
  - NFS under development
- Application Development
  - JCR Connector (JSR-170): Java API
  - SQL APIs
  - XQuery: fn:doc() and fn:collection()
  - CMIS under development (SOAP/REST API’s)
- Document Integrity: XLink, XInclude
Oracle XML DB Repository
SQL Access

• RESOURCE_VIEW
  – Contains one entry for each resource in the repository

• PATH_VIEW
  – Contains one entry for path to a resource
  – Links allow multiple paths to a single resource
  – Enables content to be organized multiple ways

• Content and Metadata exposed as an XML document

• PL/SQL packages
  • DBMS_XDB : basic CRUD operations
  • DBMS_XDB_VERSION : versioning support
Oracle XML DB Repository

SQL Operators

- **XDBURIType()**: Access content via Path
  
  ```java
  content := xdburiType('/home/SCOTT/temp.txt').getClobVal();
  ```

- **UNDER_PATH, EQUALS_PATH**
  - Used with RESOURCE_VIEW and PATH_VIEW
  
  ```sql
  select RES from RESOURCE_VIEW
  where equals_path(res,'/home/SCOTT/temp.txt') = 1
  ```

- **EQUALS_PATH()**: finds document at path
  
  ```sql
  select RES from RESOURCE_VIEW
  where equals_path(res,'/home/SCOTT') = 1
  ```

- **UNDER_PATH()**: finds documents within a folder
Oracle XML DB Repository
XQuery Operators

• `fn:doc()`:
  ```
  let $doc = fn:doc('/home/SCOTT/mydoc.xml');
  ```

• `fn:collection`
  ```
  for $doc in fn:collection('/home/SCOTT/myXMLDocs') return $doc;
  ```
PATH_VIEW in detail

<table>
<thead>
<tr>
<th>Object Type</th>
<th>VIEW Object</th>
<th>PATH_VIEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Column</td>
<td>Data Type</td>
</tr>
<tr>
<td>PATH_VIEW</td>
<td>PATH</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td></td>
<td>RES</td>
<td>XMLTYPE</td>
</tr>
<tr>
<td></td>
<td>LINK</td>
<td>XMLTYPE</td>
</tr>
<tr>
<td></td>
<td>RESID</td>
<td>RAW</td>
</tr>
</tbody>
</table>

- PATH_VIEW enables hierarchical traversal
- PATH is the path to the resource (file or folder)
- RES and LINK contain metadata about the resource and link object.
- Meta data is represented as an XMLType
Querying PATH_VIEW
Navigating the Folder Hierarchy

- Never use PATH = ‘/public/T1’ or PATH LIKE ‘/public/%’
- Use `equals_path()` or `under_path()`
  - To find a file or folder: `equals_path(res,’/public/T1’) = 1`
  - To find files in a folder: `under_path(res,’/public) =1`
Querying PATH_VIEW
Accessing File and Folder metadata

- RES and LINK contain the interesting metadata
  - RES: standard data about the resource
    - Owner, Creator, Date Created, DisplayName
  - LINK: data related to location of the resource
    - Name, Parent
- Paths are based on link Name, not DisplayName (bug in Windows)
APEX access to the XML DB Repository
Accessing XDB metadata using APEX

• XDB Repository Metadata in RES and LINK is managed as XML
• APEX does not understand XMLType very well
• Pipelined function provides APEX with access to metadata.
  – XMLTable maps content of RES and LINK to relational columns
• Nodes in RES are in the namespace
  http://xmlns.oracle.com/xdb/XDBResource.xsd
• Nodes in Link are in the namespace
  http://xmlns.oracle.com/xdb/XDBStandard
PIPEDLINE FUNCTION: DIRECTORYLIST

Enabling access to RES and LINK

```sql
cursor getFolderListing is
select PATH, RESID, RES, R.*, L.*
from PATH_VIEW,
XMLTable
(    xmlns
    (default 'http://xmlns.oracle.com/xdb/XDBResource.xsd'
    ),
    '6ES/Resource' passing RES as "RES"
columns
| IS_FOLDER | VARCHAR2(5) | PATH | '@Container',
| VERSION_ID | NUMBER(36) | PATH | '@VersionID',
| CHECKED_OUT | VARCHAR2(5) | PATH | '@IsCheckedOut',
| CREATION_DATE | TIMESTAMP(6) | PATH | 'CreationDate',
| MODIFICATION_DATE | TIMESTAMP(6) | PATH | 'ModificationDate',
| AUTHOR | VARCHAR2(128) | PATH | 'Author',
| DISPLAY_NAME | VARCHAR2(128) | PATH | 'DisplayName",
| SCHEMA_OID | RAW(16) | PATH | 'Sche0ID',
| GLOBAL_ELEMENT_ID | NUMBER(38) | PATH | 'ElNum'
) R,
XMLTable
(    xmlns
    (default 'http://xmlns.oracle.com/xdb/XDBStandard'
    ),
    '6LINK/LINK' passing LINK as "LINK"
columns
| LINK_NAME | VARCHAR2(138) | PATH | 'ChildName'
) L
where under_path(RES, 1, P_FOLDER_PATH) = 1;
```

- Cursor reads the content of the target folder
- Under_path(1) restricts results to target folder.
- XMLTable() extracts scalar values from XMLType
- Metadata for each resource returned via PIPE ROW
Repository Browser Definition

```
select nPATH, nRESID, nIS_FOLDER, nTARGET_URL, nRESOURCE_STATUS
from TABLE
    { "NFILES_APEX_SERVICES": "LISTDIRECTORY"
      :,F101_USER_ID,
      :,E1_CURRENT_FOLDER,
      :,APP_ID,
      :'FOLDERBROWSER',
      :,APP_SESSION,
      :,REQUEST,
      :,DEBUG
    }
```

- Use Query-Specific Column Names and Validate Query
- Use Generic Column Names (parse query at runtime only)

Maximum number of generic report columns:

Region Error Message
Unable to show report.
Repository Browser Application
Repository Operations
Invoking DBMS_XDB and DBMS_XDB_VERSION

- Simple abstraction layer for logging
- Basic operations mapped to DBMS_XDB methods
- Versioning enabled using DBMS_XDB_VERSION
- Extends native support for recursive operations
- PL/SQL code shared with AJAX implementation of XFILES application
Oracle XML DB repository security

- Repository security is based on Access Control Lists
  - ACLs define a user’s permissions on a file or folder
- An ACL consists of a collection of ACEs
  - Each ACE grants or revokes permissions to a principal
- Enforced using VPD (Row Level Security)
- ACLs are XML documents stored in XML DB
Example ACL
/sys/acls/bootstrap_acl.xml

```
  <acl description="Protected:Readable by PUBLIC and all privileges to OWNER"
       xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
       xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd"
       shared="true">
    <ace>
      <grant>true</grant>
      <principal>owner</principal>
    </ace>
    <ace>
      <grant>true</grant>
      <principal>XDBADMIN</principal>
    </ace>
    <ace>
      <grant>true</grant>
      <principal>PUBLIC</principal>
    </ace>
  </acl>
```
Oracle XML DB repository security

• Principals are defined as
  – A database user or role
  – An user or group defined in an Oracle LDAP server
  – A principal defined by a trusted application

• Permission include
  – read-properties, read-contents
  – update
  – resolve, link, unlink, linkto, unlinkfrom
  – read-acl, update-acl
  – dav:lock, dav:unlock
Application principals
Application Principals

- Application principals are defined by a trusted source
- Potential trusted sources include
  - LDAP Directories
  - Active Directory
  - Application managed user communities
- ACLs are evaluated in terms of the application principal, not the database user.
  - CRUD operations will be based on the permissions granted to the application principal
- Resources are created, owner, modified by application principals
Trusted Authentication model
Application Defined Principles

• Trusted applications are configured in xdbconfig.xml
  - Requires XDBADMIN role
  - Available with release 11.2.0.1.0
• Application principals can be defined statically or dynamically
• For APEX trust is based on
  - Parsing Schema, ConnectedUser and Workspace
Defining a trusted application

• First enable trust

```sql
dbms_xdb.enableCustomTrust()
```

• Add the trust scheme

```sql
dbms_xdb.addTrustScheme
(  
  NAME => ANONYMOUS_XFILES1_TRUST',  
  DESCRIPTION => 'XFILES TRUST SCHEME',  
  SESSION_USER => 'ANONYMOUS',  
  PARSING_SCHEMA => 'APEX_PARSING_SCHEMA'
)
```

• Currently needs to be enabled as follows

```sql
alter system set event='31098 trace name context forever, level 0x8000' scope=spfile
```

• Note : methods will migrate to a different package
Setting the Application Principal

• Determine the source for the Application Principal
  – User’s Email address in LDAP or WWV_FLOW_USERS
    
    ```sql
    SELECT lower(EMAIL_ADDRESS)
    INTO :F101_USER_DN
    FROM WWV_FLOW_USERS
    WHERE USER_NAME = V('APP_USER');
    ```

• Set the Application Principal before querying or updating the repository.

```sql
DECLARE
    RES BOOLEAN;
BEGIN
    RES := DBMS_XDBZ.SET_APPLICATION_PRINCIPAL(:F101_USER_DN, TRUE);
END;
```
Pluggable Repository Authentication

- Allows application principals to use WebDAV to access the repository
- Authentication scheme configured for a folder or set of folders
- Invokes a custom authentication procedure that determines whether the supplied credentials are valid
- Custom authentication procedures are defined in xdbconfig.xml
  - Requires XDBADMIN role
  - Available with release 11.2.0.1.0
Sample Custom Authentication Function

```sql
function doAuthentication(URL VARCHAR2, AUTHINFO VARCHAR2) return VARCHAR2
as
    V_USERNAME VARCHAR2(300);
    V_PASSWORD VARCHAR2(300);
Begin
    GET_USER_AND_PASSWORD(AUTHINFO, V_USERNAME, V_PASSWORD);
    if V_PASSWORD = 'oracle' then
        return '<custom_authenticate><user>
            || V_USERNAME ||
            '</user><custom_authenticate>';    
    else
        return '<custom_authenticate><error>
            || 'Invalid Password' ||
            '</error><custom_authenticate>';    
    end if;
end;
```
Configuring custom authentication

• First enable custom authentication

```sql
call dbms_xdb.enableCustomAuthentication()
```

• Add the authentication procedure

```sql
call dbms_xdb.addAuthenticationMethod
    (NAME => 'XFILES_WORKSPACE',
     description => 'Enable HTTP for users from XFILES Workspace',
     implement_schema => 'APEX_PARSING_SCHEMA',
     implement_method => 'doAuthentication',
     language => 'PL/SQL')
```
Configuring custom authentication

• Define the folders protected by the authentication scheme

```
call DBMS_XDB.ADDAUTHENTICATIONMAPPING
  (
    PATTERN => '/apexFileSystem/*',
    NAME => 'XFILES_WORKSPACE'
  )
```

• Currently needs to be enabled as follows

```
alter system set event='31098 trace name context forever, level 0x8000'
scope=spfile
```

• Note: methods will migrate to a different package
Thank you
For More Information

search.oracle.com

or

oracle.com
ORACLE IS THE INFORMATION COMPANY