29

User-Defined Repository Metadata

You can create your own metadata to associate with XML data stored in Oracle XML DB Repository.

Note:

The Oracle XML DB Repository is deprecated with Oracle Database 23ai.

- Overview of Metadata and XML
  Data that you use is often associated with additional information that is not part of the content. To process it in different ways, you can use such **metadata** to group or classify data.
- Using XML Schemas to Define Resource Metadata
  Before you can add user metadata to photo resources, you must define the structure of such metadata using XML Schema. An XML schema is created and registered for each kind (technique, category) of photo resource metadata.
- Addition, Modification, and Deletion of Resource Metadata
  You can add, update, and delete user-defined resource metadata using PL/SQL procedures in package `DBMS_XDB_REPOS`, SQL DML statements `INSERT`, `UPDATE`, and `DELETE`, or WebDAV protocol method `PROPPATCH`.
- Querying XML Schema-Based Resource Metadata
  You can use metadata column `RESID` when querying resource metadata, to join the metadata with the associated data.
- XML Image Metadata from Binary Image Metadata
  Digital cameras include image metadata as part of the image files they produce.
- Adding Non-Schema-Based Resource Metadata
  You store user-defined resource metadata that is *not* XML Schema-based as a `CLOB` instance under the `Resource` element of the associated resource.
- PL/SQL Procedures Affecting Resource Metadata
  You can use PL/SQL procedures `DBMS_XMLSCHEMA.registerSchema`, `DBMS_XDBZ.enable_hierarchy`, `DBMS_XDBZ.disable_hierarchy`, `DBMS_XDBZ.is_hierarchy_enabled`, `DBMS_XDB_REPOS.appendResourceMetadata`, `DBMS_XDB_REPOS.deleteResourceMetadata`, `DBMS_XDB_REPOS.purgeResourceMetadata`, and `DBMS_XDB_REPOS.updateResourceMetadata` to perform resource metadata operations.

Overview of Metadata and XML

Data that you use is often associated with additional information that is not part of the content. To process it in different ways, you can use such **metadata** to group or classify data.

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 be XML, but it need not be.

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 which resource information should be treated as data and which should be treated as metadata. For 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.

- Kinds of Metadata – Uses of the Term
  The term "metadata" is used in the context of XML in various ways, including XML Schema definitions, XML tags, and Oracle XML DB Repository resource information that supplements the resource content.
- 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 supplementary, related information.
- Scenario: Metadata for a Photo Collection
  A scenario used to illustrate the use of schema-based resource metadata uses metadata associated with photographic image files that are stored in repository resources. You can create any number of different kinds of metadata to be associated with the same resource.

Kinds of Metadata – Uses of the Term

The term "metadata" is used in the context of XML in various ways, including XML Schema definitions, XML tags, and Oracle XML DB Repository resource information that supplements the resource content.

In addition to resource metadata (system-defined and user-defined), the term "metadata" is sometimes used to refer to the following:

- An XML *schema* is metadata that describes a class of XML documents.
- An XML *tag* (element or attribute name) is metadata that is used to label and organize the element content or attribute value.

You can associate metadata with an XML document that is the content of a repository resource in any of these ways:

- You can add additional XML elements containing the metadata information to the resource *contents*. For example, you could wrap digital image data in an XML document that also includes elements describing the photo. In this case, the data and its metadata are associated by being in the contents of the same resource. It is up to applications to separate the two and relate them correctly.
- You can add metadata information for a particular resource to the repository as the contents of a *separate resource*. In this case, it is up to applications to treat this resource as metadata and associate it with the data.

- You can add metadata information for a resource as repository *resource metadata*. In this case, Oracle XML DB recognizes the metadata as such. Applications can *discover* this metadata by querying the repository for it. They need not be informed separately of its existence and its association with the data.

Related Topics

- Oracle XML DB Repository Resources
  Oracle XML DB Repository resources conform to the Oracle XML DB XML schema `XDBResource.xsd`. The elements in a resource include those needed to persistently store WebDAV-defined properties, such as creation date, modification date, WebDAV locks, owner, ACL, language, and character set.

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 supplementary, related information.

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

- Resource metadata that is *schema-based* 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 as part of the resource document in the resource table, `XDB.XDB$RESOURCE`.

You can take advantage of schema-based metadata, in particular, to perform efficient queries and DML operations on resources. In this chapter, you learn how to 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 learn how to add non-schema-based metadata to a resource.

You can generally use user-defined resource metadata just as you would use resource data. In particular, versioning and access control management apply.

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

Scenario: Metadata for a Photo Collection

A scenario used to illustrate the use of schema-based resource metadata uses metadata associated with photographic image files that are stored in repository resources. You can create any number of different kinds of metadata to be associated with the same resource.

For image files, examples create metadata for information about both 1) the technical aspects of a photo and 2) the photo subject or the uses to which a photo might be put. These two kinds of associated metadata are used to query photo resources.

Using XML Schemas to Define Resource Metadata

Before you can add user metadata to photo resources, you must define the structure of such metadata using XML Schema. An XML schema is created and registered for each kind (technique, category) of photo resource metadata.

See Also:

Scenario: Metadata for a Photo Collection for general information about the example user-defined metadata scenario

The XML schema in Example 29-1 defines metadata used to describe the technical aspects of a photo image file. It uses PL/SQL procedure `DBMS_XMLSCHEMA.registerSchema` to register the XML schema. To identify this schema as defining repository resource *metadata*, it uses `ENABLE_HIERARCHY_RESMETADATA` as the value for parameter `enableHierarchy`. Resource contents (data) are defined by using value `ENABLE_HIERARCHY_CONTENTS` (the default value), instead.

The properties defined in Example 29-1 are the image height, width, color depth, title, and brief description.

The XML schema in Example 29-2 defines metadata used to categorize a photo image file: to describe its content or possible uses. This simple example defines a single, general property for classification, named `Category`.

Notice that there is nothing in the XML schema definitions of metadata that restrict that information to being associated with any particular kind of data. You are free to associate any type of metadata with any type of resource. And multiple types of metadata can be associated with the same resource.

Notice, too, that the XML schema does not, by itself, define its associated data as being metadata — it is the schema *registration* that makes this characterization, through `enableHierarchy` value `ENABLE_HIERARCHY_RESMETADATA`. If the same schema were registered instead with `enableHierarchy` value `ENABLE_HIERARCHY_CONTENTS` (the default value), then it would define not metadata for resources, but resource *contents* with the same information. The same XML schema cannot be registered more than once under the same name.

Note:

By default, user metadata is stored object-relationally if it is XML schema-based and as a `CLOB` instance if non XML schema-based. You can store either as binary XML instead, by setting the `OPTIONS` parameter for XML schema registration to `REGISTER_BINARYXML`.

Example 29-1 Registering an XML Schema for Technical Photo Information

```
BEGIN
  DBMS_XMLSCHEMA.registerSchema(
    SCHEMAURL        => 'imagetechnique.xsd',
    SCHEMADOC        => '<xsd:schema targetNamespace="inamespace"
                                 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
                    xmlns:xdb="http://xmlns.oracle.com/xdb"
                    xmlns="inamespace">
          <xsd:element name="ImgTechMetadata"
                       xdb:defaultTable="IMGTECHMETADATATABLE">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="Height"      type="xsd:float"/>
                <xsd:element name="Width"       type="xsd:float"/>
                <xsd:element name="ColorDepth"  type="xsd:integer"/>
                <xsd:element name="Title"       type="xsd:string"/>
                <xsd:element name="Description" type="xsd:string"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:schema>',
    enableHierarchy => DBMS_XMLSCHEMA.ENABLE_HIERARCHY_RESMETADATA);
END;
/
```

Example 29-2 Registering an XML Schema for Photo Categorization

```
BEGIN
  DBMS_XMLSCHEMA.registerSchema(
    SCHEMAURL       => 'imagecategories.xsd',
    SCHEMADOC       => '<xsd:schema targetNamespace="cnamespace"
                                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                                   xmlns:xdb="http://xmlns.oracle.com/xdb"
                                   xmlns="cnamespace">
                         <xsd:element name="ImgCatMetadata"
                                      xdb:defaultTable="IMGCATMETADATATABLE">
                           <xsd:complexType>
                             <xsd:sequence>
                               <xsd:element name="Categories"
                                            type="CategoriesType"/>
                             </xsd:sequence>
                           </xsd:complexType>
                         </xsd:element>
                         <xsd:complexType name="CategoriesType">
                           <xsd:sequence>
                             <xsd:element name="Category" type="xsd:string"
                                          maxOccurs="unbounded"/>
                           </xsd:sequence>
                         </xsd:complexType>
                       </xsd:schema>',
    enableHierarchy => DBMS_XMLSCHEMA.ENABLE_HIERARCHY_RESMETADATA);
END;
/
```

Addition, Modification, and Deletion of Resource Metadata

You can add, update, and delete user-defined resource metadata using PL/SQL procedures in package `DBMS_XDB_REPOS`, SQL DML statements `INSERT`, `UPDATE`, and `DELETE`, or WebDAV protocol method `PROPPATCH`.

You can add, update, and delete user-defined resource metadata in any of the following ways:

- Use PL/SQL procedures in package `DBMS_XDB_REPOS`:
  - `appendResourceMetadata` – add metadata to a resource
  - `updateResourceMetadata` – modify resource metadata

  - `deleteResourceMetadata` – delete specific metadata from a resource
  - `purgeResourceMetadata` – delete *all* metadata from a resource
- Use SQL DML statements `INSERT`, `UPDATE`, and `DELETE` to update the resource directly
- Use WebDAV protocol method `PROPPATCH`

You use SQL DM statements and WebDAV method `PROPPATCH` to update or delete metadata in the same way as you add metadata. If you supply a complete `Resource` element for one of these operations, then keep in mind that each resource metadata property must be a child (not just a descendant) of element `Resource` — if you want multiple metadata elements of the same kind, you must collect them as children of a single parent metadata element. The order among such top-level user-defined resource metadata properties is unimportant and is not necessarily maintained by Oracle XML DB.

The separate PL/SQL procedures in package `DBMS_XDB_REPOS` are similar in their use. Each can be used with either XML schema-based or non-schema-based metadata. Some forms (signatures) of some of the procedures apply only to schema-based metadata. Procedures `appendResourceMetadata` and `deleteResourceMetadata` are illustrated here with examples.

- Adding Metadata Using APPENDRESOURCEMETADATA
  You can use procedure `DBMS_XDB_REPOS.appendResourceMetadata` to add user-defined metadata to resources.
- Deleting Metadata Using DELETERESOURCEMETADATA
  You can use procedure `DBMS_XDB_REPOS.deleteResourceMetadata` to delete specific metadata associated with a resource. To delete *all* of the metadata associated with a resource, you can use procedure `DBMS_XDB_REPOS.purgeResourceMetadata`.
- Adding Metadata Using SQL DML
  An alternative to using procedure `DBMS_XDB_REPOS.appendResourceMetadata` to add, update, or delete resource metadata is to update the `RESOURCE_VIEW` directly using DML statements `INSERT` and `UPDATE`.
- Adding Metadata Using WebDAV PROPPATCH
  An alternative to using procedure `DBMS_XDB_REPOS.appendResourceMetadata` to add resource metadata is to use WebDAV method `PROPPATCH`.

See Also:

Oracle Database PL/SQL Packages and Types Reference for information about the procedures in PL/SQL package `DBMS_XDB_REPOS`

Adding Metadata Using APPENDRESOURCEMETADATA

You can use procedure `DBMS_XDB_REPOS.appendResourceMetadata` to add user-defined metadata to resources.

Example 29-3 creates a photo resource and adds XML schema-based metadata of type `ImgTechMetadata` to it, recording the technical information about the photo.

Example 29-4 adds metadata of type `ImgTechMetadata` to the same resource as Example 29-3, placing the photo in several user-defined content categories.

Example 29-3 Add Metadata to a Resource – Technical Photo Information

```
DECLARE
  returnbool BOOLEAN;
BEGIN
  returnbool := DBMS_XDB_REPOS.createResource(
                  '/public/horse_with_pig.jpg',
                  bfilename('MYDIR', 'horse_with_pig.jpg'));
  DBMS_XDB_REPOS.appendResourceMetadata(
    '/public/horse_with_pig.jpg',
    XMLType('<i:ImgTechMetadata
                 xmlns:i="inamespace"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="inamespace imagetechnique.xsd">
               <Height>1024</Height>
               <Width>768</Width>
               <ColorDepth>24</ColorDepth>
               <Title>Pig Riding Horse</Title>
               <Description>Picture of a pig riding a horse on the beach,
taken outside hotel window.</Description>
             </i:ImgTechMetadata>'));
END;
/
```

Example 29-4 Add Metadata to a Resource – Photo Content Categories

```
BEGIN
  DBMS_XDB_REPOS.appendResourceMetadata(
    '/public/horse_with_pig.jpg',
    XMLType('<c:ImgCatMetadata
                 xmlns:c="cnamespace"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="cnamespace imagecategories.xsd">
               <Categories>
                 <Category>Vacation</Category>
                 <Category>Animals</Category>
                 <Category>Humor</Category>
                 <Category>2005</Category>
               </Categories>
             </c:ImgCatMetadata>'));
END;
/

PL/SQL procedure successfully completed.

SELECT * FROM imgcatmetadatatable;

SYS_NC_ROWINFO$
--------------------------------------------------------------------------------
<c:ImgCatMetadata xmlns:c="cnamespace" xmlns:xsi="http://www.w3.org/2001/XMLSche
ma-instance" xsi:schemaLocation="cnamespace imagecategories.xsd">
  <Categories>
    <Category>Vacation</Category>
    <Category>Animals</Category>
    <Category>Humor</Category>
    <Category>2005</Category>
  </Categories>
</c:ImgCatMetadata>

1 row selected.
```

Deleting Metadata Using DELETERESOURCEMETADATA

You can use procedure `DBMS_XDB_REPOS.deleteResourceMetadata` to delete specific metadata associated with a resource. To delete *all* of the metadata associated with a resource, you can use procedure `DBMS_XDB_REPOS.purgeResourceMetadata`.

Example 29-5 deletes the category metadata that was added to the photo resource in Example 29-4. By default, both the resource link (`REF`) to the metadata and the metadata table identified by that link are deleted. An optional parameter can be used to specify that only the link is to be deleted. The metadata table is then left as is but becomes unrelated to the resource. In this example, the default behavior is used.

Example 29-5 Delete Specific Metadata from a Resource

```
BEGIN
  DBMS_XDB_REPOS.deleteResourceMetadata('/public/horse_with_pig.jpg',
                                         'cnamespace',
                                         'ImgCatMetadata');
END;
/

PL/SQL procedure successfully completed.

SELECT * FROM imgcatmetadatatable;

no rows selected
```

Adding Metadata Using SQL DML

An alternative to using procedure `DBMS_XDB_REPOS.appendResourceMetadata` to add, update, or delete resource metadata is to update the `RESOURCE_VIEW` directly using DML statements `INSERT` and `UPDATE`.

Adding resource metadata in this way is illustrated by Example 29-6. It shows how to accomplish the same thing as Example 29-3 by inserting the metadata directly into `RESOURCE_VIEW` using SQL statement `UPDATE`. Other SQL DML statements may be used similarly.

Example 29-6 Adding Metadata to a Resource Using DML with RESOURCE_VIEW

```
UPDATE RESOURCE_VIEW
  SET RES =
    XMLQuery('declare namespace r = "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
              declare namespace c = "cnamespace"; (: :)
              copy $tmp := . modify insert node
               <c:ImgCatMetadata
                   xmlns:c="cnamespace"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xsi:schemaLocation="cnamespace imagecategories.xsd">
                 <Categories>
                   <Category>Vacation</Category>
                   <Category>Animals</Category>
                   <Category>Humor</Category>
                   <Category>2005</Category>
                 </Categories>
               </c:ImgCatMetadata>
              into $tmp/r:Resource
              return $tmp'
```

```
             PASSING RES
             RETURNING CONTENT)
    WHERE equals_path(RES, '/public/horse_with_pig.jpg') = 1;
/

SELECT * FROM imgcatmetadatatable;

SYS_NC_ROWINFO$
--------------------------------------------------------------------------------
<c:ImgCatMetadata xmlns:c="cnamespace" xmlns:xsi="http://www.w3.org/2001/XMLSche
ma-instance" xsi:schemaLocation="cnamespace imagecategories.xsd">
  <Categories>
    <Category>Vacation</Category>
    <Category>Animals</Category>
    <Category>Humor</Category>
    <Category>2005</Category>
  </Categories>
</c:ImgCatMetadata>

1 row selected.
```

The following query extracts the inserted metadata using `RESOURCE_VIEW`, rather than directly using metadata table `imgcatmetadatatable`. (The result is shown here pretty-printed, for clarity.)

```
SELECT XMLQuery('declare namespace r
                  = "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
                 declare namespace c
                  = "cnamespace"; (: :)
                 /r:Resource/c:ImgCatMetadata'
                PASSING RES RETURNING CONTENT)
  FROM RESOURCE_VIEW
  WHERE equals_path(RES, '/public/horse_with_pig.jpg') = 1;

XMLQUERY('DECLARENAMESPACER="HTTP://XMLNS.ORACLE.COM/XDB/XDBRESOURCE.XSD";(::)DE
--------------------------------------------------------------------------------
<c:ImgCatMetadata xmlns:c="cnamespace"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="cnamespace imagecategories.xsd">
  <Categories>
    <Category>Vacation</Category>
    <Category>Animals</Category>
    <Category>Humor</Category>
    <Category>2005</Category>
  </Categories>
</c:ImgCatMetadata>

1 row selected.
```

Adding Metadata Using WebDAV PROPPATCH

An alternative to using procedure `DBMS_XDB_REPOS.appendResourceMetadata` to add resource metadata is to use WebDAV method `PROPPATCH`.

This is illustrated in Example 29-7. You can update and delete metadata similarly.

Example 29-7 shows how to accomplish the same thing as Example 29-4 by inserting the metadata using WebDAV method `PROPPATCH`. Using appropriate tools, your application creates such a `PROPPATCH` WebDAV request and sends it to the WebDAV server for processing.

To update user-defined metadata, you proceed in the same way. To *delete* user-defined metadata, the WebDAV request is similar, but it has `D:remove` in place of `D:set`.

Example 29-7 Adding Metadata Using WebDAV PROPPATCH

```
PROPPATCH /public/horse_with_pig.jpg HTTP/1.1
Host: www.example.com
Content-Type: text/xml; charset="utf-8"
Content-Length: 609
Authorization: Basic dGRhZHhkYl9tZXRhOnRkYWR4ZGJfbWV0YQ==
Connection: close

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate xmlns:D="DAV:" xmlns:Z="http://www.w3.com/standards/z39.50/">
  <D:set>
    <D:prop>
      <c:ImgCatMetadata
          xmlns:c="cnamespace"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="cnamespace imagecategories.xsd">
        <Categories>
          <Category>Vacation</Category>
          <Category>Animals</Category>
          <Category>Humor</Category>
          <Category>2005</Category>
        </Categories>
      </c:ImgCatMetadata>
    </D:prop>
  </D:set>
</D:propertyupdate>
```

Querying XML Schema-Based Resource Metadata

You can use metadata column RESID when querying resource metadata, to join the metadata with the associated data.

When you register an XML schema using the enableHierarchy value ENABLE_HIERARCHY_RESMETADATA, an additional column, RESID, is added automatically to the XMLType tables used to store the metadata. This column stores the object identifier (OID) of the resource associated with the metadata. You can use column RESID when querying metadata, to join the metadata with the associated data.

You can query metadata in these ways:

- Query RESOURCE_VIEW for the metadata. For example:

```
SELECT count(*) FROM RESOURCE_VIEW
  WHERE
    XMLExists(
      'declare namespace r
         = "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
       declare namespace c
         = "cnamespace"; (: :)
       /r:Resource/c:ImgCatMetadata/Categories/Category[text()="Vacation"]'
      PASSING RES);

  COUNT(*)
----------
         1

1 row selected.
```

- Query the XML schema-based table for the user-defined metadata directly, and join this metadata back to the resource table, identifying which resource to select. Use column `RESID` of the metadata table to do this. For example:

  ```
  SELECT COUNT(*) FROM RESOURCE_VIEW rs, imgcatmetadatatable ct
    WHERE
      XMLExists(
        'declare namespace r
           = "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
         declare namespace c
           = "cnamespace"; (: :)
         /r:Resource/c:ImgCatMetadata/Categories/Category'
        PASSING RES)
      AND rs.RESID = ct.RESID;

    COUNT(*)
  ----------
           1

  1 row selected.
  ```

Oracle recommends querying for user-defined metadata directly, for performance reasons. Direct queries of the `RESOURCE_VIEW` alone *cannot be optimized* using XPath rewrite, because there is no way to determine whether or not target elements like `Category` are stored in the `CLOB` value or in an out-of-line table.

To improve performance further, create an index on each metadata column you intend to query.

Example 29-8 queries both kinds of photo resource metadata, retrieving the paths to the resources that are categorized as vacation photos and have the title "Pig Riding Horse".

Example 29-8 Query XML Schema-Based Resource Metadata

```
SELECT ANY_PATH
  FROM RESOURCE_VIEW rs, imgcatmetadatatable ct, imgtechmetadatatable tt
  WHERE XMLExists(
          'declare namespace r
             = "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
           declare namespace c
             = "cnamespace"; (: :)
           /r:Resource/c:ImgCatMetadata/Categories/Category[text()="Vacation"]'
          PASSING RES)
    AND XMLExists(
          'declare namespace r
             = "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
           declare namespace i
             = "inamespace"; (: :)
           /r:Resource/i:ImgTechMetadata/Title[text()="Pig Riding Horse"]'
          PASSING RES)
    AND rs.RESID = ct.RESID
    AND rs.RESID = tt.RESID;

ANY_PATH
--------------------------
/public/horse_with_pig.jpg

1 row selected.
```

XML Image Metadata from Binary Image Metadata

Digital cameras include image metadata as part of the image files they produce.

- EXIF – Exchangeable Image File Format
- IPTC-NAA IIM – International Press Telecommunications Council-Newspaper Association of America Information Interchange Model
- XMP – Extensible Metadata Platform

EXIF is the metadata standard for digital still cameras. EXIF metadata is stored in TIFF and JPEG image files. IPTC and XMP metadata is commonly embedded in image files by desktop image-processing software.

Adding Non-Schema-Based Resource Metadata

You store user-defined resource metadata that is *not* XML Schema-based as a `CLOB` instance under the `Resource` element of the associated resource.

The default XML schema for a resource has a top-level element **`any`** (declared with `maxOccurs= "unbounded"`), which admits any valid XML data as part of the resource document in the resource table, `XDB.XDB$RESOURCE`.

The following skeleton shows the structure and position of non-schema-based resource metadata:

```
<Resource xmlns="http://xmlns.oracle.com/xdb/XDBResource.xsd"
       <Owner>DESELBY</Owner>
       ... <!-- other system-defined metadata -->
       <!-- contents of the resource>
       <Contents>
         ...
       </Contents>
       <!-- User-defined metadata (appearing within different namespace) -->
       <MyOwnMetadata xmlns="http://www.example.com/custommetadata">
         <MyElement1>value1</MyElement1>
         <MyElement2>value2</MyElement2>
       </MyOwnMetadata>
     </Resource>
```

You can set and access non-schema-based resource metadata belonging to namespaces other than `XDBResource.xsd` by using any of the means described previously for accessing XML schema-based resource metadata.

Example 29-9 illustrates this for the case of SQL DML operations, adding user-defined metadata directly to the `<RESOURCE>` *document*. It shows how to add non-schema-based metadata to a resource using SQL DML.

Example 29-9   Add Non-Schema-Based Metadata to a Resource

```
DECLARE
  res BOOLEAN;
BEGIN
  res := DBMS_XDB_REPOS.createResource(
           '/public/NurseryRhyme.txt',
           bfilename('MYDIR', 'tdadxdb-xdb_repos_meta-011.txt'),
           nls_charset_id('AL32UTF8'));
  UPDATE RESOURCE_VIEW SET RES =
   XMLQuery('declare namespace r = "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
             declare namespace n = "nurserynamespace"; (: :)
             copy $tmp := . modify insert node
              <n:NurseryMetadata>
                <Author>Mother Goose</Author>
              </n:NurseryMetadata>
             into $tmp/r:Resource
             return $tmp'
            PASSING RES
            RETURNING CONTENT)
   WHERE equals_path(RES, '/public/NurseryRhyme.txt') = 1;
END;
/

PL/SQL procedure successfully completed.


SELECT XMLSerialize(DOCUMENT rs.RES AS CLOB) FROM RESOURCE_VIEW rs
  WHERE equals_path(RES, '/public/NurseryRhyme.txt') = 1;

XMLSERIALIZE(DOCUMENTRS.RESASCLOB)
----------------------------------
<Resource xmlns="http://xmlns.oracle.com/xdb/XDBResource.xsd" Hidden="false" Inv
alid="false" Container="false" CustomRslv="false" VersionHistory="false" StickyR
ef="true">
  <CreationDate>2005-05-24T13:51:48.043234</CreationDate>
  <ModificationDate>2005-05-24T13:51:48.290144</ModificationDate>
  <DisplayName>NurseryRhyme.txt</DisplayName>
  <Language>en-US</Language>
  <CharacterSet>UTF-8</CharacterSet>
  <ContentType>text/plain</ContentType>
  <RefCount>1</RefCount>
  <ACL>
    <acl description="Public:All privileges to PUBLIC" xmlns="http://xmlns.oracl
e.com/xdb/acl.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:sch
emaLocation="http://xmlns.oracle.com/xdb/acl.xsd                           http:
//xmlns.oracle.com/xdb/acl.xsd" shared="true">
      <ace>
        <principal>PUBLIC</principal>
        <grant>true</grant>
        <privilege>
          <all/>
        </privilege>
      </ace>
    </acl>
  </ACL>
  <Owner>TDADXDB_META</Owner>
  <Creator>TDADXDB_META</Creator>
  <LastModifier>TDADXDB_META</LastModifier>
  <SchemaElement>http://xmlns.oracle.com/xdb/XDBSchema.xsd#text</SchemaElement>
  <Contents>
    <text>Mary had a little lamb
```

```
Its fleece was white as snow
and everywhere that Mary went
that lamb was sure to go
</text>
  </Contents>
  <n:NurseryMetadata xmlns:n="nurserynamespace">
    <Author xmlns="">Mother Goose</Author>
  </n:NurseryMetadata>
</Resource>

1 row selected.
```

PL/SQL Procedures Affecting Resource Metadata

You can use PL/SQL procedures `DBMS_XMLSCHEMA.registerSchema`, `DBMS_XDBZ.enable_hierarchy`, `DBMS_XDBZ.disable_hierarchy`, `DBMS_XDBZ.is_hierarchy_enabled`, `DBMS_XDB_REPOS.appendResourceMetadata`, `DBMS_XDB_REPOS.deleteResourceMetadata`, `DBMS_XDB_REPOS.purgeResourceMetadata`, and `DBMS_XDB_REPOS.updateResourceMetadata` to perform resource metadata operations.

- `DBMS_XMLSCHEMA.registerSchema` – Register an XML schema. Parameter `ENABLEHIERARCHY` affects resource metadata.
- `DBMS_XDBZ.enable_hierarchy` – Enable repository support for an `XMLType` table or view. Use parameter `HIERARCHY_TYPE` with a value of `DBMS_XDBZ.ENABLE_HIERARCHY_RESMETADATA` to enable resource metadata. This adds column `RESID` to track the resource associated with the metadata.
- `DBMS_XDBZ.disable_hierarchy` – Disable all repository support for an `XMLType` table or view.
- `DBMS_XDBZ.is_hierarchy_enabled` – Tests, using parameter `HIERARCHY_TYPE`, whether the specified type of hierarchy is currently enabled for the specified `XMLType` table or view. Value `DBMS_XDBZ.IS_ENABLED_RESMETADATA` for `HIERARCHY_TYPE` tests whether resource metadata is enabled.
- `DBMS_XDB_REPOS.appendResourceMetadata` – Add metadata to a resource.
- `DBMS_XDB_REPOS.deleteResourceMetadata` – Delete specified metadata from a resource.
- `DBMS_XDB_REPOS.purgeResourceMetadata` – Delete all user-defined metadata from a resource. For schema-based resources, optional parameter `DELETE_OPTION` can be used to specify whether or not to delete the metadata information, in addition to unlinking it.
- `DBMS_XDB_REPOS.updateResourceMetadata` – Update the metadata for a resource.

See Also:

Oracle Database PL/SQL Packages and Types Reference for detailed information about these PL/SQL procedures