An Oracle White Paper
Jan 2011

The XBRL Extension to Oracle Database 11g Release 2 XML DB

ORACLE

Introduction

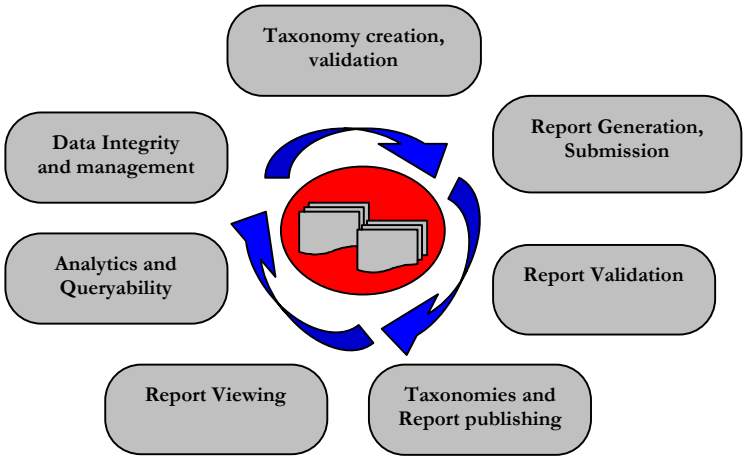
XBRL is a language for the electronic communication of business and financial data increasingly being adopted as the format for business reporting around the world. XBRL provides significant benefits in the preparation, analysis and communication of business information. At the same time, XBRL offers greater efficiency, improved accuracy and reliability to all those involved in supplying or using financial data. With growing adoption of XBRL and reports generated on a regular basis, there is a growing volume of XBRL content that needs to be stored, managed and queried efficiently.

Oracle XBRL Extension extends the Oracle XML DB to serve as a comprehensive platform for managing XBRL content. This includes an XBRL repository based on the Oracle XML DB extended with support for XBRL storage and queryability, and relational projection of XBRL data for easy integration with Oracle Business Intelligence Suite. When it is integrated with a third party XBRL processing engine, additional XBRL processing capabilities of Taxonomy design, instance publication, and instance validation can also be supported.

Architecture

XBRL Content Lifecycle

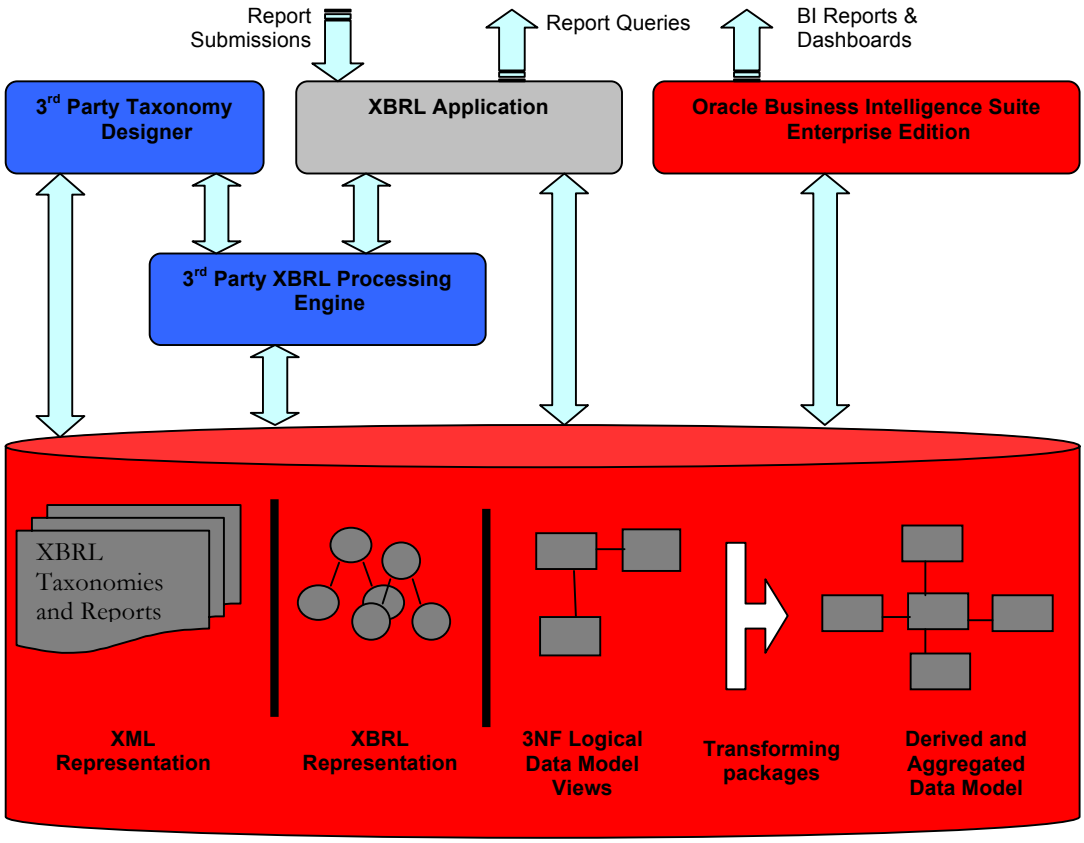
The core value proposition of XBRL is in its role as a standard for financial reporting content to allow reuse and repurposing of the content across a variety of use cases. Each use case associated with XBRL has its own requirements around XBRL processing. The use cases include filing entities generating reports for submission, Regulatory bodies validating submitted reports, and Analysts aggregating and analyzing XBRL reports.



These use cases had typically been handled by transforming the content to representations tailored for a particular use case (e.g. relational shredded forms, in-memory representations). Oracle XBRL Extension helps simplify the reuse of XBRL content across a variety of XBRL use cases and applications by providing a single repository for XBRL content that preserves the XBRL representation and semantics while also providing services to address the full spectrum of XBRL use case requirements.

XBRL Extension Architecture

Oracle XBRL Extension is comprised of a backend *XBRL Repository* based on the Oracle XML DB that provides XBRL storage and queryability along with a set of XBRL services. With its support for standard XBRL specifications, Internet protocols, and APIs, Oracle XBRL Extension allows easy integration with third party XBRL processing and business intelligence software products to satisfy XBRL processing and analysis requirements of diverse use cases.



XBRL Extension Components

XBRL Repository Storage

The XBRL Repository provides storage for XBRL content in the Oracle Database preserving its XML and document representations so that the content can be stored as is with minimum transformations. XBRL content is recognized at the time of ingestion and used to populate metadata structures that the Oracle Database uses to enforce the integrity of the XBRL content and to provide alternative representational views over it.

The XBRL repository leverages Oracle XML DB to provide XML based queryability and protocol access. A range of APIs and Internet protocols may be used for accessing the XBRL content such as Oracle OCI, JDBC, ODP.Net, SOAP, and REST. It also supports file/folder-based access to the content via WebDAV. Specialized indexing mechanisms are used to expose live relational views over the XBRL content to support integration with relational applications and SQL access. Additionally, the comprehensive strengths of the

Oracle Database can be brought to bear on the XBRL content such as security, RAC, ILM, and Partitioning.

XBRL Repository Query Processing

The XBRL Repository stores XBRL in its original XML representation while also providing relational and XBRL representational views over the content. The XBRL repository leverages Oracle XML DB to provide XML based processing directly on the documents as submitted and stored, for example for exchange.

In addition, the XBRL repository provides a relational representational view over the content by exposing a third normal form (3NF) logical data model with a set of base entities, which are physically implemented as relational views over the XBRL documents. Specialized indexing mechanisms are used to accelerate query processing of these views comparable to a physical relational implementation. The 3NF logical data model effectively provides ad-hoc queryability over the XBRL content, for example for queries of the form “find 2009 Q1 Total Revenue in Oracle’s 10-k statement” which queries the instance document only.

The XBRL repository also provides XBRL representational views over the XBRL content by exposing a set of *Network* APIs that allow reconstruction of XBRL networks from the underlying schemas, linkbases and instance documents. The XBRL networks are generated dynamically to provide real time views over the XBRL content. The XBRL networks can be used to answer *As-Filed* queries of the form “list the concepts under Total Revenue for US-GAAP in an order specified in the presentation linkbase”.

The XBRL content together with the 3NF logical data model and network APIs serve as the operational store for relational applications over the XBRL content. While much of XBRL query processing is based on querying the 3NF logical Data Model and referencing XBRL networks, XBRL data analysis is based on derived and aggregated views over the 3NF data model, such as a dimensional fact tables. To handle the full range of XBRL applications, the repository provides Transforming packages to define *derived* and *aggregated* entities.

XBRL Repository Services

The Oracle XBRL Repository also provides a suite of services to facilitate scalable XBRL operations such as diffing documents and XSLT transformations designed to minimize loading documents into memory so that these operations are efficient on large volumes of XBRL content.

Third Party XBRL Software Integration

Oracle XBRL Extension can be integrated with a separate third party XBRL processing engine that can be deployed in the mid-tier or client-tier. Third party XBRL software typically supports XBRL taxonomy design, XBRL validation, processing, and publishing based on the latest XBRL 2.1 standard. A third party XBRL software integrated with the Oracle XBRL Extension can directly operate on the stored XBRL content, reuse taxonomies in the repository, or discover taxonomies as needed. For a third party XBRL software supporting the Formula 2008 specification, complex formulas can be created to run against XBRL content stored in Oracle XBRL Extension.

Oracle Business Intelligence Suite Enterprise Edition

Oracle XBRL Extension provides relational projection of XBRL content for easy integration with Oracle Business Intelligence Enterprise Edition (OBIEE, a separate Oracle software product), which constitutes a powerful development environment for performing a wide variety of analytics, charting, reporting and publishing operations.

Oracle XBRL Extension APIs

Oracle XBRL Extension provides a basic set of APIs to serve as a foundation for XBRL applications by providing services such as storing, validating, querying, and defining derived and aggregate views over XBRL content. These services can be combined to support the wide variety of application architectures typically associated with XBRL content (e.g. a workflow engine, a custom application using BPEL and portals, or a general enterprise SOA).

XBRL Repository Storage

XBRL Storage API

The XBRL Repository Storage APIs are in the PL/SQL package DBMS_ORAXBRL. XBRL content can be loaded, deleted, or retrieved from the XBRL Repository using the APIs below. Alternatively XBRL content can also be manipulated by using WebDAV file/folder.

Name	Description
LoadSchema	This procedure will load one taxonomy schema into the XBRL repository. If the document is already present, it will replace the existing document with the given document. There is no discoverable taxonomy set (DTS) integrity check after the call.
LoadLinkbase	This procedure will load one linkbase into the XBRL repository. If the document is already present, it will replace the existing document with the given document. There is no DTS integrity check after the call.
LoadInstance	This procedure will load one instance document into the XBRL repository. It replaces the existing document If the document is already present.
bulkLoadXBRLFiles	This procedure will load a batch of files in one shot.
DTS_Files	Return the discoverable taxonomy set given a starting document.
MapPublishedLocation	This procedure should be used if protocol access is used to upload XBRL documents.
DeleteTaxonomy	This procedure will delete taxonomy, including the

	taxonomy schema and linkbases, given the location of the taxonomy schema. It will raise an error if the taxonomy is referenced by other taxonomies or instance documents in the XBRL repository, unless the *force* argument is set to 1.
DeleteLinkbase	This procedure will delete a linkbase document from the XBRL repository. It will raise error if the linkbase is referenced by other taxonomies or instance documents in the XBRL repository, unless the force argument is set to true.
DeleteInstance	This procedure will delete an instance document from the XBRL repository.
DeleteFolder	This procedure will delete all the taxonomies under the given XDB repository folder (and subfolders) forcefully without DTS integrity check.
ValidateDTSIntegrity	This function will check if all referenced taxonomy schemas and linkbases exist in the XBRL repository, and return the list of missing taxonomies. It reflects the state the XBRL repository as of the time the procedure is invoked. The result is returned as an XMLType instance.
RegisterTaxonomySchema	This procedure will register one taxonomy schema into the XBRL repository. This should be called after loadSchema. This is needed for any schema that has tuple elements and there will be queries on the tuple data.

XBRL Repository Query

The XBRL content in the XBRL Repository can be queried directly using XQuery. The XBRL Repository also provides the following:

- Relational (third normal form) views over the XBRL content for relational queryability
- Network generation APIs, to reconstruct XBRL networks
- Transforming procedures to construct derived views

Third Normal Form Logical Data Model

The XBRL Extension provides relational views over the XBRL content out of the box to provide a third normal form data model. This supports simple access to attributes of schemas, linkbases and views such as the targets of linkbases, sets of items in an instance document. These views may be queried directly, or alternative derived and aggregate views may be created over them to extract the desired representation (see step 2 in “Sample Application Flow” in this white paper)

Schema Views	ORAXBRL_XS_TARGETNSV
	ORAXBRL_XS_NSV
	ORAXBRL _XS_IMPORTNSV
	ORAXBRL_XS_LINKBASEREFV
	ORAXBRL_XS_ROLETYPEV
	ORAXBRL_XS_ARCROLETYPEV
	ORAXBRL_XS_ELEMENT
	ORAXBRL_XS_GROUPV
	ORAXBRL_XS_COMPLEXTYPESC
Linkbase Views	ORAXBRL_PRES_LINKBASE
	ORAXBRL_CALCULATION_LINKBASE
	ORAXBRL_DEFINITION_LINKBASE
	ORAXBRL_LABEL_LINKBASE
	ORAXBRL_REFERENCE_LINKBASE
Instance Views	ORAXBRL_INST_SCHEMAREFV

	ORAXBRL_INST_LINKBASEREFV
	ORAXBRL_INST_ROLEREFV
	ORAXBRL_INST_ARCROLEREFV
	ORAXBRL_INST_NSV
	ORAXBRL_INST_UNITV
	ORAXBRL_INST_CONTEXTV
	ORAXBRL_FOOTNOTES
	ORAXBRL_ SEGMENT_EXPLICITV
	ORAXBRL_ SCENARIO_ EXPLICITV
	ORAXBRL_ SEGMENT_TYPEDV
	ORAXBRL_ SCENARIO_ TYPEDV
	ORAXBRL_INST_ITEMV

Instance Network Functions: DBMS_ORAXBRLI

The instance network functions can be used to generate XBRL reports that combine taxonomy and instance data.

Name	Description
Instance_network	Return reported data organized by a base set of concept-concept relationships, such as a Presentation tree.
Multiple_instance_network	Return reported data across multiple instance documents organized by a base set of concept-concept relationships, such as a Presentation tree.

Concept Network Functions: DBMS_ORAXBRLT

The concept network functions can be used to generate XBRL taxonomy concept networks.

Name	Description
Concept_network	Return a view of a base set of concept-concept relationships, such as a Presentation tree.
Concept_roots	Return the root nodes that are descendents of a particular node in a base set tree, with labels starting with the

	entryURI specified. If no entryURI is specified, return all concept roots.
Concepts_in_tree	Return all concepts that are descendents of a particular node in a base set tree, with labels. If no entryURI is specified, return all concepts.

Transforming Procedures: DBMS_ORAXBLRV

Transforming procedures are used to generate derived views based on one of the following:

- The XBRL relational representation (i.e., the third normal form logical data model)
- The network generation APIs
- Dimensional information

Name	Description
CreateViewForConceptTree	Create a relational view for PL/SQL function *concepts_network* and *concepts_in_tree*.
CreateViewForConceptRoots	Create a relational view for PL/SQL function *concepts_roots*.
CreateViewForInstanceNetwork	Create a relational view for PL/SQL function *instance_network* and *multiple_instance_network*.
createHyperCubeFactTable	Search the hypercube network of the given primary item to find valid dimensions, then create a fact table, dimension tables, and optionally a join (as a view) between the fact table and the dimension tables.
createHyperCubeSuperFactTable	Search for primary items that include the given hypercube, then create a super fact table, dimension tables, and optionally a join (as a view) between the super fact table and the dimension tables.

Application Design

Overview

Some key considerations when defining the XBRL application architecture are:

1. Taxonomy based data model – The XBRL Extension provides a third normal form view over the XBRL content out of the box. However, depending on the taxonomies that need to be supported and the typical query/analytic operations associated with these taxonomies, the application architect must use the transforming packages provided to define derived views over the XBRL content.
2. Validation architecture – The XBRL Extension relies on other components of the application architecture to ensure that XBRL content loaded into the XBRL repository has been validated by an XBRL Processing Engine (XPE). Validation also makes sure that the Discoverable Taxonomy Set (DTS) is loaded in the XBRL repository, including the download of any missing files. Once validated, the XBRL repository can enforce the integrity of the XBRL content and its DTS. The validation itself can be done in multiple ways:
    a. Application invokes XPE immediately prior to loading: The application must invoke the XPE validate API described below prior to invoking the XBRL Repository Storage APIs for loading.
    b. Application invokes XPE asynchronously immediately after loading: The application can choose to first load the content in the XBRL repository and then invoke the XBRL processing engine to kick off a validation.
3. Deployment architecture – The XBRL Extension includes an XBRL repository based on Oracle XML DB, along with a separate third party vendor's XBRL Processing Engine and Tools deployed outside the database. The deployment architecture would need to be defined to provide sufficient processing capabilities in each tier depending on application requirements and service level agreements. Oracle XBRL Extension can be deployed with Oracle's RAC and/or partitioning options to scale up for database intensive processing.

Sample Application Flow

For purposes of illustration, a sample application for a regulatory report submission and acceptance would have the steps shown below. It uses the data in US GAAP Taxonomies, Release 2009.

1. Setup US GAAP 2009 – Download US GAAP 2009 Taxonomies (http://taxonomies.xbrl.us/us-gaap/2009/doc/XBRLUS-USGAAP-Taxonomies-2009-01-31.zip) to the working directory and unzip it. The package contains the following directory structure:

```
==========================================
GAAP2009/
|
+- schema.xml (list of schemas in us-gaap 2009)
|
+- linkbase.xml (list of linkbases in us-gaap 2009)
|
+- us-gaap/ (us-gaap 2009)
|    |
|   +- 2009/
|        |
|        + - dis/
|        |
|        + - elts/
|        |
|        + - entire/
|        |
|        + - ind/
|        |
|        + - no-gaap/
|        |
|        + - stm/
|        |
|        + - view/
|
+- Oracle/ (sample filing from Oracle Corporation, including extended taxonomy and instance)
```

schema.xml looks like the following –

```
<Upload>

 <files replace="false" httploc="http://taxonomies.xbrl.us">
```

```
  <file><name>/us-gaap/2009/dis/us-gaap-dis-acec-2009-01-
31.xsd</name></file>

  <file><name>/us-gaap/2009/dis/us-gaap-dis-ap-2009-01-
31.xsd</name></file>

…

 </files>

</Upload>
```

The `httploc` attribute specified in <`files`> informs XBRL Extension to prepend "http://taxonomies.xbrl/us" to the file names specified in `<file><name>` during the loading. This is required as the XBRL documents in US-GAAP taxonomy refer to each other with http-based absolute URI.

2. Load US GAAP 2009 taxonomies into Oracle XBRL Extension repository – Load US-GAAP 2009 base taxonomy set by using the `bulkLoadXBRLFiles` API to populate XBRL repository tables.

   First create a database directory point to *working_directory*/GAAP2009 –

   ```
   create or replace directory USGAAP2009 as
   'working_directory/GAAP2009';
   ```

   Then invoke `bulkLoadXBRLFiles` procedure –

   ```
   exec dbms_oraxbrl.bulkLoadXBRLFiles(1, 'USGAAP2009', 'schema.xml',
   null);

   exec dbms_oraxbrl.bulkLoadXBRLFiles(2, 'USGAAP2009',
   'linkbase.xml', null);
   ```

3. Query USGAAP 2009 Taxonomy – An application can directly query the relational views, invoke the *Network Generation API*, or create views from the *Network Generation API.*

   Directly query relational views -

   ```
   select count(*) from oraxbrl_xs_element;

   select count(*) from oraxbrl_calculation_linkbase;

   select count(*) from oraxbrl_pres_linkbase;
   ```

   Query with Network Generation APIs –

   ```
   select DBMS_ORAXBRLT.concepts_network('http://xbrl.us/us-gaap-
   entryPoint-std/2009-01-31', 'http://xbrl.us/us-
   gaap/role/statement/StatementOfIncome', null, 'presentationArc',
   ```

```
'http://www.xbrl.org/2003/arcrole/parent-child',
'http://www.xbrl.org/2003/role/link',
'http://www.xbrl.org/2003/arcrole/concept-label',
'http://www.xbrl.org/2003/role/label','en-US', null) from dual;
```

Create views from the Network Generation API –

```
exec dbms_oraxbrlv.createViewForConceptTree('pres_network',
'http://xbrl.us/us-gaap-entryPoint-std/2009-01-31', NULL, NULL,
'http://xbrl.us/us-gaap/role/statement/StatementOfIncome', null,
'presentationArc', 'http://www.xbrl.org/2003/arcrole/parent-
child', null, null, null,'en-US', -1);
```

4. Validate and Load New Report Submissions – New report submissions are accepted by application and used to first invoke the XBRL Processing Engine validation API and then the XBRL repository load APIs. If the report is overwriting an older report submission, the application invokes the XBRL Repository *delete* APIs which take care of deleting the old documents while maintaining integrity of the remaining content in the XBRL repository.

   Load individual filing of Oracle Corporation, with load APIs -

```
exec dbms_oraxbrl.loadSchema('/Oracle/orcl-20101130.xsd',
XMLType(BFILENAME('USGAAP2009','/Oracle/orcl-20101130.xsd'),
nls_charset_id('AL32UTF8')));

exec dbms_oraxbrl.loadLinkbase('/Oracle/orcl-20101130_pre.xml',
XMLType(BFILENAME('USGAAP','/Oracle/orcl-20101130_pre.xml'),
nls_charset_id('AL32UTF8')));

exec dbms_oraxbrl.loadLinkbase('/Oracle/orcl-20101130_def.xml',
XMLType(BFILENAME('USGAAP','/Oracle/orcl-20101130_def.xml'),
nls_charset_id('AL32UTF8')));

exec dbms_oraxbrl.loadLinkbase('/Oracle/orcl-20101130_lab.xml',
XMLType(BFILENAME('USGAAP','/Oracle/orcl-20101130_lab.xml'),
nls_charset_id('AL32UTF8')));

exec dbms_oraxbrl.loadLinkbase('/Oracle/orcl-20101130_cal.xml',
XMLType(BFILENAME('USGAAP','/Oracle/orcl-20101130_cal.xml'),
nls_charset_id('AL32UTF8')));

exec dbms_oraxbrl.loadInstance('/Oracle/orcl-20101130.xml',
XMLType(BFILENAME('USGAAP','/Oracle/orcl-20101130.xml'),
nls_charset_id('AL32UTF8')));
```

5. Setup Derived Views – Query the instance directly through relational views or by invoking *Network Generation APIs,* or generate the derived views using the *Transforming packages*, which can then be used to generate Oracle BI reports by OBIEE.

   Query directly on the relational views –

   ```
   select * from (select INSTANCE_PATH, ITEM_ID, ARC_ARCROLE,
   FOOTNOTE_ROLE, FOOTNOTE_TITLE, FOOTNOTE_LANG, FOOTNOTE_CONTENT
   from oraxbrl_footnotes order by instance_path, item_id) where
   rownum < 10;
   ```

   Query through Network Generation API –

   ```
   select
   DBMS_ORAXBRLI.Instance_Network('http://xbrl.oracle.com/20090831','
   0001341439','01-JUN-10', '30-NOV-10',
   'http://xbrl.oracle.com/20090831/role/StockholdersEquity', null,
   'presentationArc', 'http://www.xbrl.org/2003/arcrole/parent-
   child', null, null, null, 'en-US', 1) as res from dual;
   ```

   Query using the Transforming package –

   ```
   exec dbms_oraxbrlv.createHyperCubeSuperFactTable('orcl',
   '0001341439', 'http://xbrl.oracle.com/20090831',
   'http://xbrl.us/us-gaap/2009-01-31', 'StatementTable',
   'http://xbrl.oracle.com/20090831/CondensedConsolidatedBalanceSheet
   s', 'segment',
   'http://xbrl.oracle.com/20090831/CondensedConsolidatedBalanceSheet
   s');
   ```

   This will create a fact table `user_STATEMENTTABLE` and dimension tables `user_STATEMENTCLASSOFSTOCKA`, `user_ STATEMENTSCENARIOAXIS`, and `ORCL`.

6. Integrate with OBIEE – Oracle Business Intelligence Enterprise Edition can be used to produce business intelligence reports by using the derived views generated in the previous step.

7. Drop Individual Filings – Drop the individual filing with *delete* APIs.

   ```
   exec dbms_oraxbrl.deleteinstance('/Oracle/orcl-20101130.xml');

   exec dbms_oraxbrl.deletetaxonomy('/Oracle/orcl-20101130.xsd');
   ```

   The execution order is important here. If `deleteTaxonomy` is invoked before `deleteInstance`, an error will be raised because the taxonomy is still referred by the instance. This is enforced by the document integrity.

8. Drop USGAAP 2009 – Drop US-GAAP 2009 with deleteFolder API.

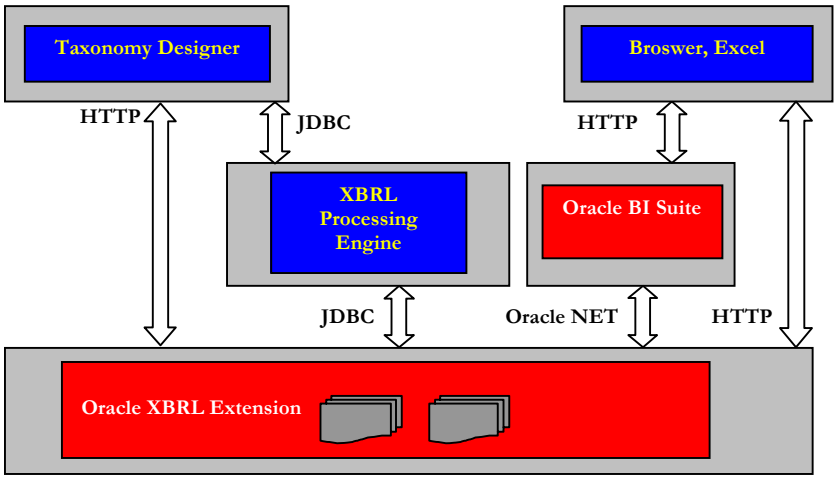
```
exec
dbms_oraxbrl.deletefolder('http://taxonomies.xbrl.us');
```

Please note that "http://taxnomies.xbrl.us" is used as the folder. That is due to the `httploc` attribute in specified in schema.xml and linkbase.xml, as described in step 0.

Deployment

The Oracle XBRL Extension is comprised of the foregoing functional pieces deployed in traditional three-tier architecture with the XBRL repository in the database tier, an XBRL processing engine from a third party vendor and potentially the Oracle BI Suite as a separate instance in the midtier, and a set of tools on the client desktop.

These functional pieces can be combined using a prepackaged XBRL workflow suite like the Enterprise Application Suite from UBmatrix. Combining the functional pieces together in alternative integration environment with portals and BPEL support can create custom deployments.



The deployment of these requires the usual evaluations of the application requirements and service level agreements to determine the scale of processing needed in the database tier, XBRL processing engine and the Oracle BI Suite. Query and analytic intensive applications may require scaling up the Oracle BI Suite and the database tier by using Oracle RAC and partitioning options. To satisfy data security requirements, Oracle Audit Vault should be considered for deployment. Finally, XBRL processing intensive applications may require scaling up the XBRL processing capabilities.

Conclusion

With XBRL growing in adoption and with a steady increase in the volume of XBRL content, the Oracle XBRL Extension provides a comprehensive platform for managing the life cycle of large volumes of XBRL content.

Feature Matrix

The set of features is listed below.

XBRL Repository
● *Storage* – Database native XBRL storage
● *Document Integrity* – Database enforcement of integrity based on XBRL rules
● *Protocol Access* – File/Folder view of content
● *XML Queryability* – XML queryability based on XBRL semantics
● *3NF Logical Data Model* – SQL queryability and relational application integration
● *Transforming Packages* – Transforming Packages for Derived views
● *Scalable XBRL Services* – Report and network generation, transformations
XBRL Processing (requires separate 3rd party vendor software)
● *Validation* – validate any XBRL document instance or taxonomy
● *Processing* – process and transform any XBRL document
● *Formulas* - Execute 2008 formulas
● *Dimensions* – Online analytics based on XBRL dimensions (explicit, typed)
XBRL Tools (requires separate 3rd party vendor and Oracle software)
● *Taxonomy Designer* – Design Phase IDE to create, edit, validate taxonomies
● *OBIEE* – Out-of-the-box integration with OBIEE

Oracle XBRL Extension
Jan 2011

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

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

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

0109