Oracle® Database

XBRL Extension Developer's Guide

12*c* Release 1 (12.1)

E17070-06

December 2015

This manual describes XBRL Extension to Oracle XML DB.

ORACLE®

Oracle Database XBRL Extension Developer's Guide, 12*c* Release 1 (12.1)

E17070-06

Primary Author: Drew Adams

Contributor: The Oracle Database 12*c* documentation is dedicated to Mark Townsend, who was an inspiration to all who worked on this release.

Contributor: Sriram Krishnamurthy, Ying Lu, Qin Yu

Contents

3 APIs – XBRL Extension to Oracle XML DB

4 Administering XBRL Extension to Oracle XML DB

5 Installing XBRL Extension to Oracle XML DB

Index

List of Examples

List of Figures

List of Tables

Preface

This manual describes XBRL Extension to Oracle XML DB.

Audience

This manual is intended for developers building XBRL applications.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following Oracle resources:

- *Oracle XML DB Developer's Guide*
- *Oracle XML Developer's Kit Programmer's Guide*

To download free release notes, installation documentation, white papers, or other collateral material, visit Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

http://www.oracle.com/technetwork/community/join/overview/index.html

If you have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

http://www.oracle.com/technetwork/documentation/

For additional information, see:

- http://www.w3.org/TR/xml/ – XML (language)
- http://www.xml.com/pub/a/98/10/guide0.html – XML introduction
- http://www.w3.org/XML/Schema – XML Schema

- http://www.w3.org/2001/XMLSchema – XML Schema
- http://www.w3.org/TR/xmlschema-0/ – XML Schema: primer
- http://www.w3.org/TR/xmlschema-1/ – XML Schema: structures
- http://www.w3.org/TR/xmlschema-2/ – XML Schema: data types
- http://www.oasis-open.org/cover/schemas.html – XML Schema
- http://www.xml.com/pub/a/2000/11/29/schemas/part1.html – XML Schema
- http://xml.coverpages.org/xmlMediaMIME.html – media/MIME types
- http://www.w3.org/TR/xptr/ – XPointer
- http://www.w3.org/TR/xpath – XPath 1.0
- http://www.w3.org/TR/xpath20/ – XPath 2.0
- http://www.zvon.org/xxl/XPathTutorial/General/examples.html – XPath
- *XML In a Nutshell*, by Elliotte Rusty Harold and W. Scott Means, O'Reilly, January 2001, http://www.oreilly.com/catalog/xmlnut/chapter/ch09.html
- http://www.w3.org/TR/2002/NOTE-unicode-xml-20020218/ – Unicode in XML
- http://www.w3.org/TR/xml-names/ – XML namespaces
- http://www.w3.org/TR/xml-infoset/ – information sets
- http://www.w3.org/DOM/ – Document Object Model (DOM)
- http://www.w3.org/TR/xslt – XSLT
- http://www.w3.org/TR/xsl – XSL
- http://www.oasis-open.org/cover/xsl.html – XSL
- http://www.zvon.org/xxl/XSLTutorial/Books/Book1/index.html – XSL
- http://www.w3.org/2002/ws/Activity.html – Web services
- http://www.ietf.org/rfc/rfc959.txt – RFC 959: FTP Protocol Specification
- ISO/IEC 13249-2:2000, Information technology - Database languages - SQL Multimedia and Application Packages - Part 2: Full-Text, International Organization For Standardization, 2000
- http://java.sun.com/xml/tutorial_intro.html – XML and Java

Note: Throughout this manual, "**XML Schema**" refers to the XML Schema 1.0 recommendation, http://www.w3.org/XML/Schema.

Placeholders in Oracle Database XBRL Extension Developer's Guide

The following placeholders are used in this book, in particular in Chapter 5, "Installing XBRL Extension to Oracle XML DB" and Chapter 4, "Administering XBRL Extension to Oracle XML DB".

- *sys_pass* – System password.
- *xb_sys_pass* – Password for database user XBRLSYS, which is the user that creates and administers all XBRL repositories.
- *xb_sys_ts* – A tablespace for user XBRLSYS. For Oracle Database 11g Release 2 (11.2.0.2) or later, the tablespace must use automatic segment space management.

- *xb_sys_tmp_ts* – A temporary tablespace for user XBRLSYS.
- *xb_rep* – Name of an XBRL repository, which is also a database user name. (See "Creating an XBRL Repository" on page 5-4.)
- *xb_rep_pass* – Password for database user *xb_rep*.
- *xb_rep_ts* – A tablespace for XBRL repository *xb_rep*. For Oracle Database 11g Release 2 (11.2.0.2) or later, the tablespace must use automatic segment space management.
- *xb_rep_idx_ts* – Tablespace for the XMLIndex index storage tables for XBRL repository *xb_rep*.
- *xb_rep_tmp_ts* – A temporary tablespace for XBRL repository *xb_rep*.
- *xb_protocols* – Whether or not to use Oracle XML DB Repository to provide protocol access. TRUE means use it; FALSE means do not use it. If XBRL repository *xb_rep* is likely to contain more than 100,000 documents, then use FALSE for best performance.
- *obiee_home* – Directory where Oracle Business Intelligence Suite Enterprise Edition (OBIEE) is installed.
- *obieedata_home* – Directory where OBIEE data is stored.
- *oracle_client* – Name of your Oracle client for OBIEE. For example, **OracleClient11g_home1**.
- *oracle_client_dir* – Directory where your Oracle client for OBIEE is installed.

> **See Also:** *Oracle Database Administrator's Guide* for information about automatic segment space management

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Code Examples

The code examples in this book are for illustration only. In many cases, however, you can copy and paste parts of examples and run them in your environment.

Pretty Printing of XML Data

To promote readability, especially of lengthy or complex XML data, output is sometimes shown pretty-printed (formatted) in code examples.

Reminder About Case Sensitivity

When examining the examples in this book, keep in mind the following:

- SQL is case-insensitive, but names in SQL code are implicitly uppercase, unless you enclose them in double-quotes.
- XML is case-sensitive. You must refer to SQL names in XML code using the correct case: uppercase SQL names must be written as uppercase.

For example, if you create a table named `my_table` in SQL without using double-quotes, then you must refer to it in XML code as `"MY_TABLE"`.

Changes in This Release for XBRL Extension to Oracle XML DB

This preface describes the new features and functionality, enhancements, APIs, and product integration support added to XBRL Extension to Oracle XML DB.

It also describes the deprecation of certain XBRL Extension to Oracle XML DB constructs.

Topics:

XBRL Extension to Oracle XML DB 11g Release 3 (11.2.0.2.2) New Features

The following features are new in XBRL Extension to Oracle XML DB 11*g* Release 3 (11.2.0.2.2).

- You can use a different tablespace for the `XMLIndex` index storage tables.
- You can partition the table that stores your XBRL instance documents. And you can store these partitions in different tablespaces.

See Also: Chapter 4, "Administering XBRL Extension to Oracle XML DB"

XBRL Extension to Oracle XML DB 11g Release 2 (11.2.0.2.1) New Features

The following features are new in XBRL Extension to Oracle XML DB 11*g* Release 2 (11.2.0.2.1).

- Table (view) names for a star schema generated by functions and procedures in PL/SQL package `DBMS_ORAXBRLV` are now unique across multiple invocations.
- New routines createStarSchemaFromFact and createStarSchemaFromHC have been added to PL/SQL package `DBMS_ORAXBRLV` to retrieve the list of table (view) names of a generated star schema.
- New routine DTS_filelist has been added to PL/SQL package `DBMS_ORAXBRL` to retrieve a discoverable taxonomy set (DTS).

XBRL Extension to Oracle XML DB 11g Release 2 (11.2.0.2.1) Deprecated Constructs

The following constructs are *deprecated* in XBRL Extension to Oracle XML DB 11*g* Release 2 (11.2.0.2.1). They are still supported for backward compatibility, but Oracle recommends that you do not use them in new applications. They will be desupported in a future release.

- Procedure `createFactTable`. Use `createHyperCubeFactTable` and `createStarSchemaFromFact` instead.
- Procedure `createSuperFactTable`. Use `createHyperCubeSuperFactTable` and `createStarSchemaFromHC` instead.

1

Overview of XBRL Extension to Oracle XML DB

This chapter introduces XBRL Extension to Oracle XML DB. It covers these topics:

- XBRL and XBRL Extension to Oracle XML DB
- Overview of XBRL Extension to Oracle XML DB Features
- Architecture

XBRL and XBRL Extension to Oracle XML DB

XBRL (eXtensible Business Reporting Language) is a language for the electronic communication of business and financial data. It is used as the format for business reporting around the world. XBRL Extension to Oracle XML DB extends Oracle Database to serve as a comprehensive platform for managing XBRL content.

XBRL provides significant benefits in the preparation, analysis, and communication of business information. XBRL offers greater efficiency and improved accuracy and reliability for all those involved in supplying or using financial data. With growing adoption of XBRL, and with financial reports being generated on a regular basis, there is a growing volume of XBRL content to be stored, managed, and queried efficiently.

XBRL Extension to Oracle XML DB helps you manage XBRL content. It lets you create multiple XBRL repositories and project XBRL data relationally or query it in various ways. It can help you improve operations on aggregated business and financial reports such as extraction, transformation, and loading (ETL); business intelligence (BI); and online analytical processing (OLAP).

Overview of XBRL Extension to Oracle XML DB Features

XBRL Extension to Oracle XML DB provides the following features.

- Native database storage of XBRL data.
- Database enforcement of integrity, based on XBRL rules.
- Access to XBRL content using APIs and protocols, including WebDAV, which provides a files-and-folders view of content.
- Ability to query XML data using XBRL semantics.
- Relational representation of XBRL content. Ability to expose XBRL content to relational applications and SQL queries.
- PL/SQL transforming procedures that generate derived XBRL views based on XBRL relational representations, network generation APIs, or dimensional information.

- Scalable XBRL services: reports, network generation, transformations.
- Online analysis based on XBRL dimensions, both explicit and typed.
- Integration with Oracle Business Intelligence Suite Enterprise Edition (OBIEE).

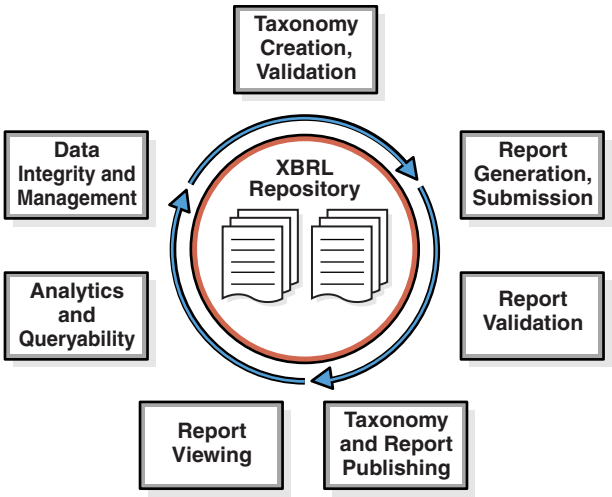
Architecture

This section describes the architecture of XBRL Extension to Oracle XML DB.

XBRL Content Lifecycle

The typical lifecycle of XBRL content is depicted in Figure 1–1. XBRL lets you reuse and repurpose content across a variety of use cases. These use cases include filing organizations generating financial reports for submission, regulatory bodies validating submitted financial reports, and analysts aggregating and analyzing financial reports.

Figure 1–1 XBRL Content Lifecycle



The use cases have typically been handled by transforming the content to representations that are tailored for each use case (decomposed, relational forms; in-memory representations; and so on). XBRL Extension to Oracle XML DB 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 breadth of the use case requirements.

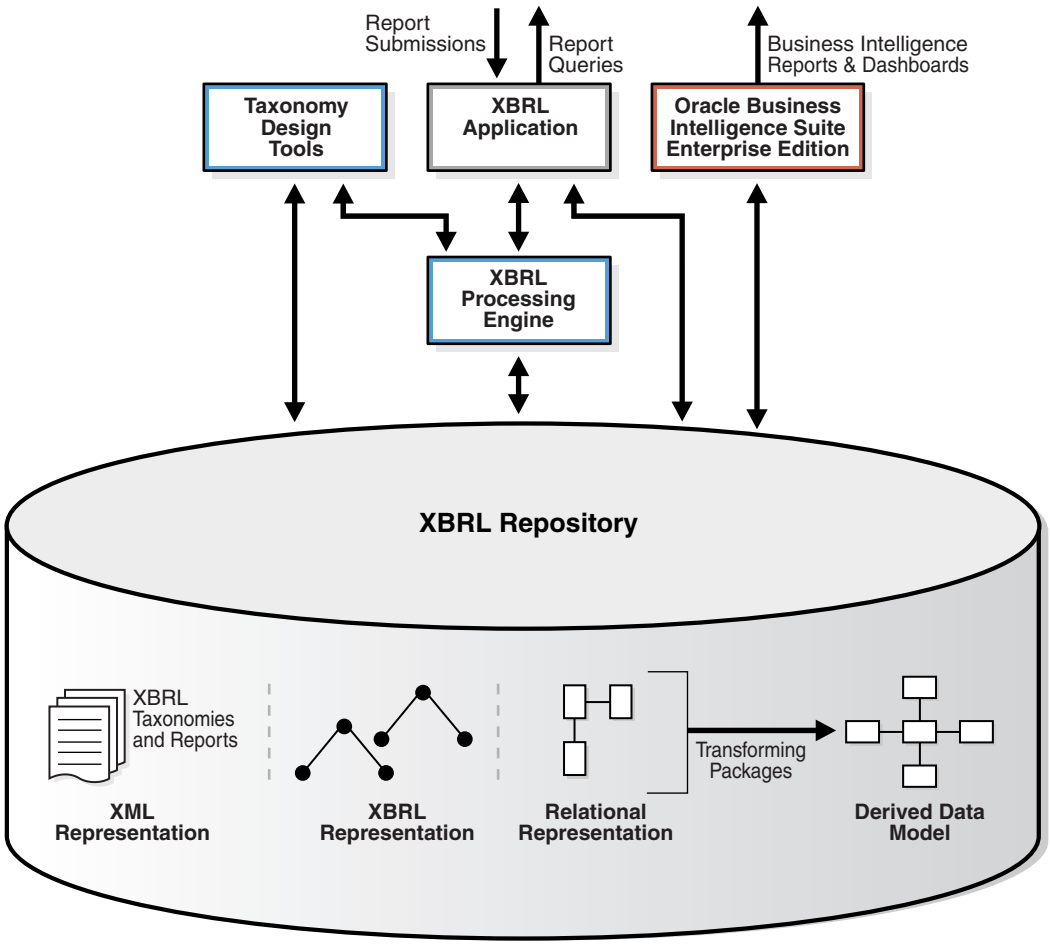
Architecture of XBRL Extension to Oracle XML DB

The architecture of XBRL Extension to Oracle XML DB is shown in Figure 1–2. XBRL Extension to Oracle XML DB is composed of the following:

- One or more back-end XBRL repositories based on Oracle Database, which provide XBRL storage and queryability with a set of XBRL-specific services
- An external XBRL processing engine (XPE)

XBRL Extension to Oracle XML DB integrates easily with Oracle Business Intelligence Suite Enterprise Edition (OBIEE) for analytics and with interactive development environments (IDEs) and design tools for creating and editing XBRL taxonomies.

Figure 1–2 Architecture of XBRL Extension to Oracle XML DB



Components of XBRL Extension to Oracle XML DB

This section provides an overview of the main components of XBRL Extension to Oracle XML DB, which provide storage, querying, services, processing, integration with Oracle Business Intelligence Suite Enterprise Edition (OBIEE), and an integrated taxonomy design environment.

XBRL Extension to Oracle XML DB: Storage

XBRL Extension to Oracle XML DB provides storage for XBRL content in Oracle Database, preserving its XML and document representations so that the content can be stored with minimal transformation. XBRL content is recognized at the time of ingestion and populates metadata structures that the database uses to enforce the integrity of the XBRL content and to provide alternative representational views of it.

XBRL Extension to Oracle XML DB leverages Oracle XML DB to provide XML-based queryability and protocol access. You can use a range of APIs and protocols to access XBRL content, including Oracle OCI, JDBC, ODP.NET, and Web services (SOAP and REST). XBRL Extension to Oracle XML DB also supports file-and-folder access to XBRL content, using WebDAV. Specialized indexing mechanisms expose live relational views of XBRL content, to support integration with relational applications and access using SQL. The traditional strengths of Oracle Database, including Oracle Real Application Clusters (Oracle RAC), Information Lifecycle Management (ILM), and partitioning, can be brought to bear on XBRL content.

XBRL Extension to Oracle XML DB: Querying

XBRL Extension to Oracle XML DB stores XBRL content in its original XML representation. It leverages Oracle XML DB to provide XML-based processing directly on the documents as submitted and stored, for example, for exchange purposes.

XBRL Extension to Oracle XML DB provides a relational representational view of your XBRL content, by exposing a third normal form logical data model with a set of base entities. Specialized indexing mechanisms are used to speed up these database views to make them comparable to a physical relational implementation.

The relational representation effectively provides ad-hoc queryability of your XBRL content. For example, you can query an instance document to find the 2010 total first-quarter revenue in an Oracle 10-k statement. Such a query does not reference taxonomies; it accesses only tables of instance documents.

The XBRL repository also provides XBRL representational views over your 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 and provide real time views over the XBRL content. You can use the XBRL networks to answer as-filed queries such as listing the concepts under the category Total Revenue for US-GAAP in an order specified in the presentation linkbase.

Together, the XBRL content, the relational representation, and the network APIs serve as the operational store for relational applications that access the XBRL content. While much of XBRL query processing is based on querying the relational representation while referencing XBRL networks, XBRL analysis is based on derived views, such as dimensional fact tables, over the relational representation. To handle the full range of XBRL applications, XBRL Extension to Oracle XML DB provides transforming packages to define derived entities.

XBRL Extension to Oracle XML DB: Services

XBRL Extension to Oracle XML DB provides services that facilitate scalable XBRL-based operations, including comparing and transforming XML documents. Such operations are designed to minimize loading of documents into memory.

XBRL Processing

XBRL Extension to Oracle XML DB requires a third-party XBRL processing engine (XPE) that is deployed in either the client or the middle tier. The XPE must operate directly on XBRL content in the XBRL repository to reuse taxonomies and discover missing taxonomies.

See Also:

`http://www.xbrl.org/Specification/formula/REC-2009-06-22/formula-REC-2009-06-22.html` for information about XBRL Formula 1.0

Oracle Business Intelligence Suite Enterprise Edition

XBRL Extension to Oracle XML DB provides relational projection of XBRL data, for easy integration with Oracle Business Intelligence Suite Enterprise Edition (OBIEE). OBIEE is not included with XBRL Extension to Oracle XML DB; you must procure it separately. OBIEE is a powerful development environment that helps you perform a wide variety of analytical, charting, reporting, and publishing operations.

2

Using XBRL Extension to Oracle XML DB

This chapter explains how to use XBRL Extension to Oracle XML DB to create XBRL applications.

This chapter covers these topics:

- XBRL Application Design
- Deployment of XBRL Extension to Oracle XML DB

> **See Also:** Chapter 3, "APIs – XBRL Extension to Oracle XML DB" for information about the APIs referred to in this chapter

XBRL Application Design

This section provides general information about building XBRL applications.

Overview of XBRL Application Architecture

The following are some things to consider when defining an XBRL application architecture:

- Taxonomy-based data model

  XBRL Extension to Oracle XML DB provides a relational representation of your XBRL content. This is a third-normal form (3NF) view of the XBRL data. However, depending on the taxonomies to be supported and the typical query and analysis operations associated with these taxonomies, consider using transforming procedures to define derived views over the XBRL content.

  > **See Also:** "Transforming Procedures: DBMS_ORAXBRLV" on page 3-18

- Validation Architecture

  An XBRL repository relies on your application architecture to ensure that XBRL content has been validated by an external XBRL processing engine (XPE). Validation also ensures that the relevant discoverable taxonomy set (DTS) is loaded into the XBRL repository and that any missing files are downloaded. After XBRL content is validated, the XBRL repository can enforce the integrity of the content and its DTS. The validation itself can be done in either of these ways:

  - Your application can invoke the XPE validation API just before you load the XBRL content into the XBRL repository.
  - Your application can invoke XPE asynchronously after you load XBRL content into the XBRL repository.

- Deployment architecture

  XBRL Extension to Oracle XML DB includes one or more XBRL repositories, which are based on Oracle Database. You can use Oracle Real Application Clusters (Oracle RAC) or partitioning with a deployed XBRL repository. XBRL Extension to Oracle XML DB also requires an external XBRL processing engine that is deployed outside the database. In addition, you can optionally deploy XBRL Extension to Oracle XML DB together with Oracle Business Intelligence Suite Enterprise Edition (OBIEE) and tools that help with XBRL taxonomy design. The deployment architecture must provide sufficient processing capabilities in each tier, depending on application requirements and service-level agreements.

  See Also: "Deployment of XBRL Extension to Oracle XML DB" on page 2-11

Using a Shared Database Schema for Enterprise User Security

The user whose name is the same as the XBRL repository, *xb_rep*, has general access to the repository. In general you do not want to give application users this much access. Typically application users are allowed only to load and delete documents.

For security reasons Oracle recommends that you follow the enterprise user security model, as described in *Oracle Database Enterprise User Security Administrator's Guide*. You create a database user that is granted only the access that you want to provide application users, and then your application users share that database user to access the database.

You restrict access for the shared database user (and hence application users) by defining a PL/SQL package under user *xb_rep* that has a restricted set of APIs, typically only APIs to load and delete XBRL repository documents.

Example 2–1 illustrates this.

- *xb_rep* is the database user whose name is the same as the XBRL repository; *xb_rep_pass* is the corresponding password.
- *xb_app* is the shared database user; *xb_app_pass* is the corresponding password.
- *xb_app_pkg* is the package that provides the APIs you want to make available for application users.
- *xb_rep* grants privilege `EXECUTE` to *xb_app* for package *xb_app_pkg*.

Example 2–1 Creating a PL/SQL Package for Application Users

```
CONNECT xb_rep/xb_rep_pass

CREATE OR REPLACE PACKAGE xb_app_pkg AS
  PROCEDURE loadSchema(schemaLoc VARCHAR2,
                       schemaDoc XMLType,
                       valid     PLS_INTEGER DEFAULT NULL,
                       auxLoc    VARCHAR2 DEFAULT NULL);
  PROCEDURE loadLinkBase(linkbaseLoc VARCHAR2,
                         linkbaseDoc XMLType,
                         valid       PLS_INTEGER DEFAULT NULL,
                         auxLoc      VARCHAR2 DEFAULT NULL);
  PROCEDURE loadInstance(instanceLoc VARCHAR2,
                         instanceDoc XMLType,
                         valid       PLS_INTEGER DEFAULT NULL,
                         auxLoc      VARCHAR2 DEFAULT NULL);
  PROCEDURE loadAuxDocument(auxDocPath    VARCHAR2,
```

```
                            auxilliaryDoc XMLType,
                            valid         PLS_INTEGER DEFAULT NULL,
                            auxLoc        VARCHAR2 DEFAULT NULL);
  PROCEDURE deleteTaxonomy(schemaTargetNS VARCHAR2,
                           force          PLS_INTEGER DEFAULT 0);
  PROCEDURE deleteLinkbase(linkbaseLoc VARCHAR2,
                           force PLS_INTEGER DEFAULT 0);
  PROCEDURE deleteInstance(instanceLoc VARCHAR2);
  PROCEDURE deleteFolder(folder VARCHAR2);
  PROCEDURE bulkLoadXBRLFiles(operation NUMBER,
                              directory VARCHAR2,
                              filelist  VARCHAR2,
                              target    VARCHAR2);
END xb_app_pkg;
/
CREATE OR REPLACE PACKAGE BODY xb_app_pkg AS
  PROCEDURE loadSchema(schemaLoc VARCHAR2,
                       schemaDoc XMLType,
                       valid     PLS_INTEGER DEFAULT NULL,
                       auxLoc    VARCHAR2 DEFAULT NULL) IS
    BEGIN
      DBMS_ORAXBRL.loadSchema(schemaLoc, schemaDoc, valid, auxLoc);
    EXCEPTION
      WHEN OTHERS THEN
        RAISE;
    END loadSchema;

-- Define the other procedure bodies similarly.
...

GRANT EXECUTE ON xb_app_pkg TO xb_app;
```

An application user can then connect to the database as shared database user *xb_app* and invoke procedures in the package *xb_app_pkg*, as shown in Example 2–2.

Example 2–2 Invoking an Application User Procedure

```
CONNECT xb_app/xb_app_pass
EXEC xb_rep.xb_app_pkg.loadSchema(
       '/boa/boa-20061231.xsd',
       XMLType(bfilename('USGAAP', '/boa/boa-20061231.xsd'),
               nls_charset_id('AL32UTF8')));
```

Creating a Parameter File for Bulk-Loading a Set of XBRL Documents

You can bulk-load a set of XBRL documents of the same type, whether that type is schema, link-base, or instance, by invoking procedure `DBMA_ORAXBRL.bulkLoadFiles`, passing it a parameter file that lists the documents to load.

This section explains how to create such a parameter file. See "Building and Using a Sample XBRL Application: USGAAP 2008" on page 2-4 for a specific example of this using the taxonomy set USGAAP 2008 and an instance filing from Bank of America.

Proceed as follows to create a bulk-load parameter file:

1. Find the lowest file-system directory that contains all of the documents that you want to load. Call this the **base directory** for the set of documents. Create the (empty) parameter file in this directory.
2. In the parameter file, create an `Upload` element with a `files` element child.

3. For *each* document to be loaded, create a `file` element as a child of element `files`, and a `name` element as child of element `file`. For the text node of element `name`, use the file name for the document to be loaded, *relative to the base directory*.
4. If the documents to be loaded refer to their base taxonomy using an HTTP-based URL, then you must specify the network location of the documents using optional attribute `httploc` of element `files`.

   For the value of attribute `httploc` you can start with the value of any of these attributes:

   - Attribute `schemaLocation` from an `import` element of an extension schema.
   - Attribute `xlink:href` from a `link:schemaRef` element of an XBRL instance document or an extension linkbase file.
   - Attribute `href` from a `locator` element of a linkbase file.

   Given such a value, for attribute `httploc` you use everything up to but not including the slash that precedes the path. More precisely:

   - An HTTP-based URL has the following syntax, where `username:password@` and `:port` are optional (and `HTTPS` can be used in place of `HTTP`):

     `HTTP://username:password@domain:port``/path?query_string#anchor`

   - As the value of attribute `httploc`, you use the part of the URL shown in bold, `HTTP://`*`username`*`:`*`password`*`@`*`domain`*`:`*`port`*, or more typically just `HTTP://`*`domain`*.

   For example, if you start with an `href` attribute of a `locator` element in a linkbase file, and if that attribute value is `http://www.xbrl.org/2003/XLink`, then the value of attribute `httploc` is `http://www.xbrl.org`.

Building and Using a Sample XBRL Application: USGAAP 2008

This section presents the steps to build and use a sample application for a regulatory report submission and acceptance. It uses the taxonomy set USGAAP 2008 and an instance filing from Bank of America.

> **See Also:** "Installing the Sample XBRL Repository and GAAP Demo" on page 5-6

Step 1: Set Up USGAAP

Copy files `schema.xml` and `linkbase.xml` from directory `xbrl_xdb/XBRLScripts` to a working directory, for example, `gaap2008`. See "Directory xbrl_xdb" on page 5-6 for information about these files.

Download USGAAP 2008 and the Bank of America filing from the following URLs:

- `http://www.sec.gov/cgi-bin/goodbye.cgi?taxonomies.xbrl.us/us-gaap/1.0/doc/XBRLUSGAAPTaxonomies-2008-03-31.zip`
- `http://xbrl.us/us-gaap/instance/1.0/boa-20061231/boa-20061231.zip`

Copy those two zip files to your working directory and unzip them. Extract the USGAAP documents to directory `gaap2008/us-gaap/1.0/` and the Bank of America filings to directory `gaap2008/boa/`. Example 2–3 shows the expected directory structure.

Example 2–3 Structure of Directory gaap2008

```
gaap2008/
|
+- schema.xml (list of schemas in us-gaap 2008)
+- linkbase.xml (list of linkbases in us-gaap 2008)
|
+- us-gaap/ (us-gaap 2008)
| |
| +- 1.0/
|    |
|    + - dis/
|    + - elts/
|    + - ind/
|    + - no-gaap/
|    + - stm/
|    + - test/
|
+- boa/ (filing from Bank of America, including extended taxonomy and instance)
  |
  + - boa-20061231.xsd
  + - boa-20061231_cal.xml
  + - boa-20061231_pre.xml
  + - boa-20061231_def.xml
  + - boa-20061231_lab.xml
  + - boa-20061231_XML.xml
```

Example 2–4 shows the format of parameter file `schema.xml` that is used to bulk-load the documents.

Example 2–4 Format of Bulk-Load Parameter File schema.xml

```
<Upload>
  <files httploc="http://xbrl.us" commitnum=10>
    <file>
      <name>/us-gaap/1.0/dis/us-gaap-dis-acec-2008-03-31.xsd</name>
    </file>
    <file>
      <name>/us-gaap/1.0/dis/us-gaap-dis-ap-2008-03-31.xsd</name>
    </file>
    …
  </files>
</Upload>
```

The values of the `name` elements are the names of the files to be loaded, *relative to the directory* where the files are located (their base directory).

Attribute `httploc` of element `files` specifies that the files are to be loaded using HTTP. It tells XBRL Extension to Oracle XML DB to prepend the URL prefix `http://xbrl.us` to each of the file names when loading. This is required because the XBRL documents in the US-GAAP taxonomy refer to each other using HTTP-based URLs. See "Creating a Parameter File for Bulk-Loading a Set of XBRL Documents" on page 2-3.

Attribute `commitnum` is optional. It specifies an automatic commit frequency for bulk-loading *instance* documents: a `COMMIT` operation is performed after every *N* instance-document insertions, where *N* is the commitnum value. The default value is 1000. This attribute has no effect on bulk-loading other types of documents: for non-instance documents a `COMMIT` is performed after loading each document.

Step 2: Load USGAAP

Load the base taxonomy set, USGAAP 2008, using the `bulkLoadXBRLFiles` API.

First, connect as the user whose name is the same as the XBRL repository, *xb_rep*.

```
CONNECT xb_rep/xb_rep_pass
```

Then create a database directory that points to *working_directory*/gaap2008:

```
CREATE OR REPLACE DIRECTORY usgaap AS 'working_directory/gaap2008';
```

Then invoke the `bulkLoadXBRLFiles` procedure which populates the system tables:

```
EXEC DBMS_ORAXBRL.bulkLoadXBRLFiles(1, 'USGAAP', 'schema.xml', NULL);
EXEC DBMS_ORAXBRL.bulkLoadXBRLFiles(2, 'USGAAP', 'linkbase.xml', NULL);
```

Step 3: Query USGAAP Taxonomy

The application can directly query the relational views, invoke the network generation API, or create views using the network generation API. See "Instance Network Functions: DBMS_ORAXBRLI" on page 3-13 for details

Query relational views:

```
SELECT count(*) FROM oraxbrl_xs_element;
SELECT count(*) FROM oraxbrl_calculation_linkbase;
SELECT count(*) FROM oraxbrl_pres_linkbase;
```

Query using network generation APIs:

```
SELECT DBMS_ORAXBRLT.concepts_network(
         'http://xbrl.us/us-gaap-entryPoint-std/2008-03-31',
         '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 using the network generation API:

```
EXEC DBMS_ORAXBRLV.createViewForConceptTree(
  'pres_network',
  'http://xbrl.us/us-gaap-entryPoint-std/2008-03-31',
  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);
```

Step 4: Validate and Load New Report Submissions

Applications accept new report submissions and invoke the XBRL repository load APIs to load the new reports. If a report overwrites an older report submission, the

application invokes the XBRL repository deletion APIs to delete the old documents while maintaining the integrity of the remaining content in the XBRL repository.

Load the Bank of America filing:

```
EXEC DBMS_ORAXBRL.loadSchema('/boa/boa-20061231.xsd',
                             XMLType(BFILENAME('USGAAP', '/boa/boa-20061231.xsd'),
                             nls_charset_id('AL32UTF8')));

EXEC DBMS_ORAXBRL.loadLinkbase('/boa/boa-20061231_pre.xml',
                               XMLType(BFILENAME('USGAAP',
                                                 '/boa/boa-20061231_pre.xml'),
                               nls_charset_id('AL32UTF8')));

EXEC DBMS_ORAXBRL.loadLinkbase('/boa/boa-20061231_def.xml',
                               XMLType(BFILENAME('USGAAP',
                                                 '/boa/boa-20061231_def.xml'),
                               nls_charset_id('AL32UTF8')));

EXEC DBMS_ORAXBRL.loadLinkbase('/boa/boa-20061231_lab.xml',
                               XMLType(BFILENAME('USGAAP',
                                                 '/boa/boa-20061231_lab.xml'),
                               nls_charset_id('AL32UTF8')));

EXEC DBMS_ORAXBRL.loadLinkbase('/boa/boa-20061231_cal.xml',
                               XMLType(BFILENAME('USGAAP',
                                                 '/boa/boa-20061231_cal.xml'),
                               nls_charset_id('AL32UTF8')));

EXEC DBMS_ORAXBRL.loadInstance('/boa/boa-20061231_XML.xml',
                               XMLType(BFILENAME('USGAAP',
                                                 '/boa/boa-20061231_XML.xml'),
                               nls_charset_id('AL32UTF8')));
```

Step 5: Query the Instance or Generate Derived Views

An application can query the instance using XBRL relational views or by invoking network generation APIs. Or it can generate derived views using transforming procedures, and then use Oracle Business Intelligence Suite Enterprise Edition (OBIEE) to generate business intelligence reports.

See Also:

- "Instance Network Functions: DBMS_ORAXBRLI" on page 3-13
- "Transforming Procedures: DBMS_ORAXBRLV" on page 3-18

Query directly using XBRL 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 using a network generation API:

```
SELECT DBMS_ORAXBRLI.Instance_Network(
         'http://xbrl.boa.com/2006-12-31',
         '0000070858',
         DATE'2005-01-01',
         DATE'2006-01-01',
```

```
        'http://xbrl.boa.com/2006-12-31/ext/IncomeStatement',
        NULL,
        'presentationArc',
        'http://www.xbrl.org/2003/arcrole/parent-child',
        NULL,
        NULL,
        NULL,
       'en-US',
       1)
   FROM DUAL;
```

Query using a transforming procedure. This SQL statement creates fact table `user_STATEMENTTABLE`, dimension table `user_ STATEMENTEQUITYCOMPONENT`, and view `boa`.

```
EXEC DBMS_ORAXBRLV.createHyperCubeSuperFactTable(
       'boa',
       '0000070858',
       'http://xbrl.boa.com/2006-12-31',
       'http://xbrl.us/us-gaap/2008-03-31',
       'StatementTable',
       'http://xbrl.boa.com/2006-12-31/ext/StockholdersEquity',
       'segment',
       'http://xbrl.boa.com/2006-12-31/ext/StockholdersEquity');
```

See Also: "Demo-BIFiles" on page 5-7 for a sample business intelligence report for Bank of America that uses the derived views generated here

Step 6: Integrate with Oracle Business Intelligence Suite Enterprise Edition

Integrate with Oracle Business Intelligence Suite Enterprise Edition (OBIEE). There is a sample business intelligence report for Bank of America that uses the derived views generated in "Step 5: Query the Instance or Generate Derived Views" on page 2-7.

See Also: "Integration with Oracle Business Intelligence Suite" on page 5-7

Step 7: Drop Individual Filings

Drop the individual filing using the deletion APIs.

```
EXEC DBMS_ORAXBRL.deleteInstance('/boa/boa-20061231_XML.xml');
EXEC DBMS_ORAXBRL.deleteTaxonomy('/boa/boa-20061231.xsd');
```

The order of execution is important here. Invoking `deleteTaxonomy` before `deleteInstance` raises an error, because the taxonomy is still referred to by the instance.

Step 8: Drop USGAAP

Drop USGAAP 2008:

```
EXEC DBMS_ORAXBRL.deleteFolder('http://xbrl.us');
```

`http://xbrl.us` is used as the folder. This is because attribute `httploc` is specified in `schema.xml` and `linkbase.xml`, as described in "Step 1: Set Up USGAAP".

Building and Using a Sample XBRL Application: Tuples

This section illustrates the steps needed to build and use a sample XBRL application with tuples.

Step 1: Create XBRL Standard Schema Files for the Tuple Application

Download the following XBRL standard schema files from http://www.xbrl.org, and copy them to a working directory, *xb_cd*.

- http://www.xbrl.org/2003/xlink-2003-12-31.xsd
- http://www.xbrl.org/2003/xl-2003-12-31.xsd
- http://www.xbrl.org/2003/xbrl-linkbase-2003-12-31.xsd
- http://www.xbrl.org/2003/xbrl-instance-2003-12-31.xsd
- http://www.xbrl.org/2006/ref-2006-02-27.xsd
- http://www.xbrl.org/2004/ref-2004-08-10.xsd
- http://www.xbrl.org/2005/xbrldt-2005.xsd
- http://www.xbrl.org/2006/xbrldi-2006.xsd

Edit the downloaded schema files to update attribute `schemaLocation` of element `import`. Prefix *each* `schemaLocation` value with `http://www.xbrl.org/200`***N***`/`, where ***N*** corresponds to the downloaded schema.

For example, for schema `xl-200`**3**`-12-31.xsd` the original `schemaLocation` value is `"xlink-200`**3**`-12-31.xsd"`. You must change it to `"http://www.xbrl.org/200`**3**`/xlink-200`**3**`-12-31.xsd"`.

Step 2: Register XBRL Standard Schemas for the Tuple Application

Run SQL script `$ORACLE_HOME/rdbms/xbrl_xdb/XBRLScripts/xbrlregschema.sql` to register the XBRL standard schemas.

Step 3. Register the Sample XBRL Taxonomy for the Tuple Application

Set events 31098 and 31156, load the taxonomy (XML schema), and register it with Oracle XML DB. The taxonomy to be registered is `oraclexbrltupledemo.xsd`. The data table for tuple element `ComplexItems` is `complexitemstab`.

Example 2–5 Register Sample XBRL Taxonomy for Tuple Application

```
ALTER SESSION SET EVENTS = '31098 trace name context forever';
ALTER SESSION SET EVENTS = '31156 trace name context forever, level 1';

DECLARE
  xmlSchema XMLType;
BEGIN
  xmlSchema := XMLType(bfilename('DEMODIR', 'oraclexbrltupledemo.xsd'),
                       nls_charset_id('AL32UTF8'));
  DBMS_ORAXBRL.loadSchema('http://www.oracle.com/oraclexbrltupledemo.xsd',
                          xmlSchema, NULL, NULL);
  DBMS_ORAXBRL.registerTaxonomySchema(
    'http://www.oracle.com/oraclexbrltupledemo.xsd',
    XMLType('<schemaAnnotation>
               <element>
                 <elementName>ComplexItems</elementName>
                 <defaultTableName>complexitemstab</defaultTableName>
               </element>
```

```
            </schemaAnnotation>'));
END;
/

-- Show the tuple elements and corresponding tables created during registration.

SELECT table_name, element_name FROM oraxbrl_tuple_tables ORDER BY table_name;
```

Step 4: Create Tuple Elements and Corresponding Data Tables

A given taxonomy can contain multiple top-level tuple elements: occurrences of element `element` that have an attribute `substitutionGroup` with value `xbrli:tuple`. For each such top-level tuple element you can create a tuple data table. Example 2–5 creates one such a tuple data table at the time of taxonomy registration.

You can use procedure `createTupleDataTable` to create tuple data tables for additional top-level tuple elements. Alternatively you can register the taxonomy without creating any tuple data tables and then use `createTupleDataTable` to create them. In that case, you would pass `NULL` as the second argument to procedure `registerTaxonomySchema`. Example 2–6 illustrates this—it presumes that Example 2–5 was used with `NULL` as the second argument in place of the `XMLType` construction.

Example 2–6 Using DBMS_ORAXBRL.CREATETUPLEDATATABLE

```
DBMS_ORAXBRL.createTupleDataTable(
  'http://www.oracle.com/oraclexbrltupledemo.xsd',
  'complexitemstab', 'ComplexItems');
```

Step 5: Load an XBRL Instance with Tuple Data

Example 2–7 uses procedure `loadInstance` to load the tuple data for element `ComplexItems` from file-system file `oraclexbrltupledemo-inst.xml` into a tuple data table.

Example 2–7 Loading an XBRL Instance with Tuple Data

```
EXEC DBMS_ORAXBRL.loadInstance(
  'http://www.oracle.com/oraclexbrltupledemo-inst.xml',
  XMLType(bfilename('DEMODIR', 'oraclexbrltupledemo-inst.xml'),
  nls_charset_id('AL32UTF8')));
```

Example 2–8 shows how to query this tuple data. It queries elements `ComplexItems/DescriptionContent` and `ComplexItems/AmountTotal`, retrieving their text nodes and values of attribute `contextRef`.

Example 2–8 Querying Tuple Data

```
SELECT t1.contextRef, t1.description, t2.totalvalue
  FROM ora$xbrlpath p,
       complexitemstab t,
       XMLTable(XMLNamespaces(DEFAULT 'http://www.oracle.com/oraclexbrltupledemo'),
                '/ComplexItems' PASSING t.tupledata
                COLUMNS description PATH 'DescriptionContent/text()',
                        contextRef  VARCHAR2(4000) PATH 'DescriptionContent/@contextRef',
                        totalamount XMLType PATH 'AmountTotal') t1,
       XMLTable(XMLNamespaces(DEFAULT 'http://www.oracle.com/oraclexbrltupledemo'),
                '/AmountTotal' PASSING t1.totalamount
                COLUMNS totalvalue PATH 'text()',
                        contextRef PATH '@contextRef') t2
  WHERE p.DOCOID  = t.DOCOID
```

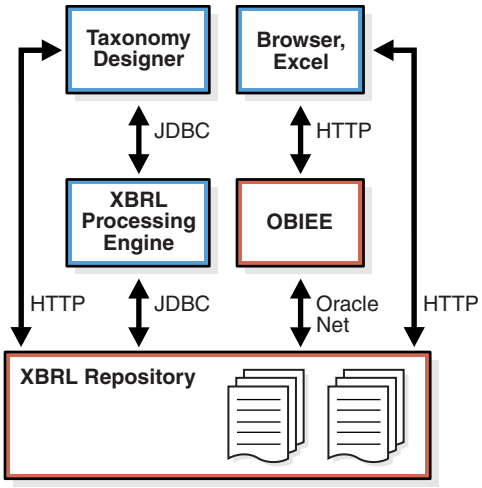
```
AND p.DOCPATH = 'http://www.oracle.com/oraclexbrltupledemo-inst.xml'
AND t1.contextRef = t2.contextRef;
```

Deployment of XBRL Extension to Oracle XML DB

You deploy XBRL Extension to Oracle XML DB in a traditional three-tier architecture, with XBRL repositories in the database-tier, the XBRL processing engine and optionally Oracle BI Suite as a separate instance in the mid-tier, and a set of tools in the client-tier (desktop). This is illustrated in Figure 2–1.

These functional pieces can be combined using an XBRL workflow suite from a third-party vendor. Alternatively, you can create a custom deployment by combining functional pieces using portals and Business Process Execution Language (BPEL)

Figure 2–1 XBRL Extension to Oracle XML DB: Deployment



Deployment requires the usual evaluations of application requirements and service-level agreements to determine the scale of processing needed in the database, XBRL processing engine, and Oracle BI Suite. Query- and analysis-intensive applications can require scaling up Oracle BI Suite and the database tier using Oracle Real Application Clusters (Oracle RAC) or partitioning. XBRL processing-intensive applications can require scaling up the XBRL processing capabilities.

3

APIs – XBRL Extension to Oracle XML DB

This chapter describes the application programming interfaces (APIs) provided by XBRL Extension to Oracle XML DB. It covers these topics:

- XBRL Repository Storage API: DBMS_ORAXBRL
- XBRL Repository Query

Several of the examples in this chapter refer to the downloadable demonstration examples. See "Installing the Sample XBRL Repository and GAAP Demo" on page 5-6.

Note:

- There are two versions of some of the functions described here. The versions have the same name, except that one name has "2" appended to it. The version whose name ends in "2" returns an instance of data type `ORAXBRL_CONCEPTLIST` (or `ORAXBRL_ITEMLIST`, in the case of `multiple_instance_network2`). The version whose name has no "2" appended to it returns an `XMLType` instance.
- When, for a given parameter, no default value is declared in the signature of a procedure or function, and no default value for that parameter is mentioned in the text, it means that an error is raised if the argument is `NULL`.

XBRL Repository Storage API: DBMS_ORAXBRL

The XBRL repository storage APIs are in PL/SQL package `DBMS_ORAXBRL`. An application can use this package to load, delete, or retrieve XBRL content from an XBRL repository. (Alternatively, you can manipulate XBRL content using WebDAV files and folders.) An application can also use this package to register a taxonomy schema or investigate discoverable taxonomy set (DTS) information.

Note: The procedures in package `DBMS_ORAXBRL` that load a single document into the XBRL repository have a parameter whose value is an absolute URI that is used as the *base location* for any relative URIs that are found in the document to be loaded (and in any documents referenced from that document). More precisely, this base location is used to interpret relative URIs that are the values of attributes `xlink:href` and `schemaLocation`.

It is your responsibility to ensure that the location parameter you use specifies the proper base location for any such relative URIs.

Table 3–1 DBMS_ORAXBRL Repository Storage APIs

Name	Description
`bulkLoadXBRLFiles`	Load a set of XBRL files.
`createTupleDataTable`	Create an object-relational `XMLType` storage table for a tuple element.
`deleteAuxDocument`	Delete an auxiliary document from the XBRL repository.
`deleteFolder`	Forcefully (without DTS integrity check) delete all of the taxonomies under the given XBRL repository folder.
`deleteInstance`	Delete an instance document from the XBRL repository.
`deleteLinkbase`	Delete a linkbase document from the XBRL repository. Raise an error if the linkbase is referenced by other taxonomies or instance documents in the XBRL repository, unless the `force` argument is `1` or greater.
`deleteTaxonomy`	Delete a taxonomy, including its schema and linkbases, given the location of the taxonomy schema. Raise an error if the taxonomy is referenced by other taxonomies or instance documents in the XBRL repository, unless the `force` argument is `1` or greater.
`dropTupleDataTable`	Drop the `XMLType` storage table for a tuple element.
`DTS_files`	Return a discoverable taxonomy set (DTS), given a starting document.
DTS_filelist	Return the discoverable taxonomy set (DTS) for a given URI.
`getAuxDocForRepoPath`	Return an auxiliary document that is associated with a document in the XBRL repository.
`getDocument`	Return a document that is in the XBRL repository.
`isDocPathValid`	Return the XBRL validity of a document in the XBRL repository.
`loadAuxDocument`	Load an auxiliary document into the XBRL repository.
`loadInstance`	Load one instance document into the XBRL repository. If the document is present in the repository, replace it.
`loadLinkbase`	Load one linkbase into an XBRL repository. If the document is present in the repository, replace it. DTS integrity is not checked after the call.
`loadSchema`	Load one taxonomy schema into an XBRL repository. If the document is present in the repository, replace it. Discoverable taxonomy set (DTS) integrity is not checked after the call.

Table 3–1 (Cont.) DBMS_ORAXBRL Repository Storage APIs

Name	Description
mapPublishedLocation	Apply a published location (a URL) recursively to all files and folders under the specified Oracle XML DB Repository folder. Invoke this procedure after uploading XBRL documents if you use protocols to upload.
registerTaxonomySchema	Register one taxonomy schema with the XBRL repository. Invoke this after loadSchema. This is needed for any schema that has tuple elements.
updateDocValidity	Update the validity status of a document and associate an auxiliary document with it.
validateDTSIntegrity, validateDTSIntegrity2	Check whether all referenced taxonomy schemas and linkbases exist in the XBRL repository. Return the list of taxonomies as an XML document, indicating which are missing. The return value reflects the state of the repository at the time the procedure is invoked.

The detailed API information is given the following sections.

bulkLoadXBRLFiles

Upload a set of files. Before invoking this procedure, create a database directory.

```
PROCEDURE bulkLoadXBRLFiles(operation NUMBER,
                            directory VARCHAR2,
                            filelist  VARCHAR2,
                            target    VARCHAR2);
```

Table 3–2 DBMS_ORAXBRL.BULKLOADXBRLFILES Parameters

Parameter	Description
operation	Type of operation to be carried out: ▪ 1: Load a taxonomy schema. ▪ 2: Load a linkbase. ▪ 3: Load an XBRL instance. ▪ 4: Load an auxiliary document. ▪ 5: Register a taxonomy schema. ▪ 6: Insert instance tuple data into a tuple data table. Load the corresponding instance file first, if not already loaded. Raise an error if NULL.
directory	Database directory where the files reside. Raise an error if NULL.
filelist	XML document listing the file names to be loaded. Raise an error if NULL.
target	Folder location in Oracle XML DB Repository where documents will be uploaded. If NULL, then the upload folder is taken from the locations in filelist.

Note: If many documents are to be bulk-loaded, consider using multiple sessions so they are loaded in parallel.

createTupleDataTable

Create a tuple data table. The table has these columns:

- `docid` (`RAW16`) – An XBRL instance document identifier.
- `tupledata` (`XMLType` stored object-relationally) – Instance data for a tuple element of an XBRL schema.

```
PROCEDURE createTupleDataTable(schemaLoc  VARCHAR2,
                               table_name VARCHAR2,
                               elem_name  VARCHAR2);
```

Does nothing if any of the arguments is `NULL`.

Table 3–3 DBMS_ORAXBRL.CREATETUPLEDATA Parameters

Parameter	Description
`schemaLoc`	Location in the XBRL repository of the XBRL schema on which the tuple data table will be based. This must be an absolute URI, by which other documents can refer to this taxonomy schema. Do nothing if `NULL`.
`table_name`	Name of the `XMLType` table to be created. Do nothing if `NULL`.
`elem_name`	Name of the tuple element. Do nothing if `NULL`.

deleteAuxDocument

Delete an auxiliary document from the XBRL repository.

```
PROCEDURE deleteAuxDocument(auxDocPath VARCHAR2);
```

Table 3–4 DBMS_ORXBRL.DELETEAUXDOCUMENT Parameters

Parameter	Description
`auxDocPath`	Location in the XBRL repository of the auxiliary document to be deleted.

deleteFolder

Forcefully (without DTS integrity check) delete all of the taxonomies under the given XBRL repository folder.

```
PROCEDURE deleteFolder(folder VARCHAR2);
```

Table 3–5 DBMS_ORXBRL.DELETEFOLDER Parameters

Parameter	Description
`folder`	XBRL repository folder that contains the taxonomies to be deleted. Raise an error if `NULL`. If you use Oracle XML DB Repository to provide protocol access, then folder can alternatively name a folder in Oracle XML DB Repository. To delete the root (top-level) folder, use `deleteFolder('/')`.

deleteInstance

Delete an XBRL instance document from the XBRL repository. Delete the tuple instance data from the tuple data table, if present.

```
PROCEDURE deleteInstance(instanceLoc VARCHAR2);
```

Table 3–6 DBMS_ORXBRL.DELETEINSTANCE Parameters

Parameter	Description
`instanceLoc`	The `instanceLoc` value that was specified for `loadInstance` when this instance document was loaded. Raise an error if `NULL`.

deleteLinkbase

Delete a linkbase document from the XBRL repository. Raise an error if the linkbase is referenced by other taxonomies or instance documents in the XBRL repository, unless the `force` argument is `1` or greater.

```
PROCEDURE deleteLinkbase(linkbaseLoc VARCHAR2, force PLS_INTEGER DEFAULT 0);
```

Table 3–7 DBMS_ORXBRL.DELETELINKBASE Parameters

Parameter	Description
`linkbaseLoc`	The `linkbaseLoc` value that was specified for `loadLinkbase` when this linkbase was loaded. Raise an error if `NULL`.
`force`	`1` or greater: Delete without first performing a DTS integrity check. `0` or less: Raise an error if the check fails. Raise an error if `NULL`.

deleteTaxonomy

Delete a taxonomy, including the taxonomy schema and linkbases, given the target namespace of the taxonomy schema. Raise an error if the taxonomy is referenced by other taxonomies or instance documents in the XBRL repository, unless argument `force` is `1` or greater. If the taxonomy schema was registered, then delete the schema.

```
PROCEDURE deleteTaxonomy(repopath VARCHAR2, force PLS_INTEGER DEFAULT 0);
```

Table 3–8 DBMS_ORXBRL.DELETETAXONOMY Parameters

Parameter	Description
`repopath`	Location in the XBRL repository of an XBRL taxonomy schema. This must be an absolute URI, by which other documents can refer to this taxonomy schema. Raise an error if `NULL`.
`force`	`1` or greater: Delete without first performing a DTS integrity check. `0` or less: Delete if a DTS integrity check succeeds. Raise an error if the check fails. Raise an error if `NULL`.

dropTupleDataTable

Drop the tuple data for a tuple element.

```
PROCEDURE dropTupleDataTable(repopath   VARCHAR2,
                             table_name VARCHAR2,
                             elem_name  VARCHAR2,
                             force      IN PLS_INTEGER DEFAULT 0);
```

Table 3–9 DBMS_ORXBRL.DROPTUPLETABLE Parameters

Parameter	Description
`repopath`	Location in the XBRL repository of an XBRL schema. This must be an absolute URI, by which other documents can refer to this taxonomy schema. Do nothing if `NULL`.
`table_name`	Name of the tuple data table to be dropped. Do nothing if `NULL`.
`elem_name`	Name of the tuple element. Do nothing if `NULL`.
`force`	If `1` or greater then drop the table. If `0` or less then: ■ If the table is empty then drop it. ■ Otherwise, raise an error.

DTS_files

Return the discoverable taxonomy set (DTS) for a given URI. The first time you invoke this procedure, it builds a cache for the DTS. Subsequent calls simply return the cached result. Also, calculations of the DTS that are implicit in other procedures and queries use the cached result instead of recomputing the DTS. Use of the cache can make network queries run faster.

If a DTS is expected to be large for a given entry URI, then invoke this procedure after loading all documents and before network generation. As a guideline, if a DTS lists more than 100 documents, then it will take at least a second to compute the DTS. For example, for entry URI `'http://xbrl.us/us-gaap-entryPoint-std/2008-03-31'`, the DTS contains 600 entries, so it takes several seconds to compute the list.

```
PROCEDURE DTS_files(entryURI IN VARCHAR2);
```

Table 3–10 DBMS_ORXBRL.DTS_FILES Parameters

Parameter	Description
`entryURI`	Entry URI for the required DTS. Raise an error if `NULL`.

DTS_filelist

Return the discoverable taxonomy set (DTS) for a given URI. This function behaves similarly to procedure `DTS_files`, but:

- It always returns the cached result, if found. If not found, it builds (and caches) the DTS.
- It returns `NULL` instead of raising an error if the argument is `NULL`.

```
FUNCTION  DTS_filelist(entryURI IN VARCHAR2) RETURN ORAXBRL_DTSURLLIST;
```

Table 3–11 DBMS_ORXBRL.DTS_FILELIST Parameters

Parameter	Description
`entryURI`	Entry URI for the required DTS. Return `NULL` if `NULL`.

getAuxDocForRepoPath

Return the auxiliary document that is associated with the document that is at a specified XBRL repository location.

```
FUNCTION getAuxDocForRepopath(repoPath VARCHAR2) RETURN XMLType;
```

Table 3–12 DBMS_ORAXBRL.GETAUXDOCFORREPOPATH Parameters

Parameter	Description
repoPath	Location in the XBRL repository of the document that is associated with the auxiliary document to retrieve.

getDocument

Return the document that is at a specified XBRL repository location.

```
FUNCTION getDocument(repoPath VARCHAR2) RETURN XMLType;
```

Table 3–13 DBMS_ORAXBRL.GETDOCUMENT Parameters

Parameter	Description
repoPath	Location in the XBRL repository of the document to retrieve.

isDocPathValid

Return the XBRL validity of a document in the XBRL repository. 1 means the document is valid; 0 means it is invalid.

```
FUNCTION isDocPathValid(repoPath VARCHAR2) RETURN PLS_INTEGER;
```

Table 3–14 DBMS_ORAXBRL.ISDOCPATHVALID Parameters

Parameter	Description
repoPath	Location in the XBRL repository of the document.

loadAuxDocument

Load an auxiliary document into the XBRL repository at a specified location.

```
PROCEDURE loadAuxDocument(auxDocPath VARCHAR2, auxiliaryDoc XMLType);
```

Table 3–15 DBMS_ORAXBRL.LOADAUXDOCUMENT Parameters

Parameter	Description
auxDocPath	Location in the XBRL repository to load the auxiliary document auxiliaryDoc.
auxiliaryDoc	Auxiliary document to load into repository location auxDocPath.

loadInstance

Load one instance document into the XBRL repository. For a top-level tuple element whose taxonomy schema is registered and whose tuple data table has been created, insert the instance tuple data into the tuple data table.

```
PROCEDURE loadInstance(instanceLoc VARCHAR2,
                       instanceDoc XMLType,
                       valid       IN PLS_INTEGER DEFAULT NULL,
                       auxLoc      VARCHAR2 DEFAULT NULL);
```

Table 3–16 DBMS_ORXBRL.LOADINSTANCE Parameters

Parameter	Description
instanceLoc	Location in the XBRL repository of instance document instanceDoc. This must be an absolute URI. Raise an error if NULL. Any relative URIs used as the value of attribute xlink:href in document instanceDoc are interpreted relative to location instanceLoc. You must ensure that instanceLoc is the proper base location for any such relative URIs.
instanceDoc	XBRL instance document. Raise an error if NULL.
valid	Validity of document instanceDoc. 1: valid, 0: invalid.
auxLoc	XBRL repository location of an XBRL auxiliary document that is mapped to instanceDoc.

loadLinkbase

Load one XBRL linkbase into the XBRL repository. DTS integrity is not checked after the call.

```
PROCEDURE loadLinkbase(linkbaseLoc VARCHAR2,
                       linkbaseDoc XMLType,
                       valid       IN PLS_INTEGER DEFAULT NULL,
                       auxLoc      VARCHAR2 DEFAULT NULL);
```

Table 3–17 DBMS_ORXBRL.LOADLINKBASE Parameters

Parameter	Description
linkbaseLoc	Location in the XBRL repository of linkbase document doc. This must be an absolute URI, by which other documents can refer to this linkbase. Raise an error if NULL. Any relative URIs used as the value of attribute xlink:href in document linkbaseDoc are interpreted relative to location linkbaseLoc. You must ensure that linkbaseLoc is the proper base location for any such relative URIs.
linkbaseDoc	XBRL linkbase document. Raise an error if NULL.
valid	Validity of document linkbaseDoc. 1: valid, 0: invalid.
auxLoc	XBRL repository location of an XBRL auxiliary document that is mapped to linkbaseDoc.

loadSchema

Load one taxonomy schema into an XBRL repository. DTS integrity is not checked after the call.

```
PROCEDURE loadSchema(schemaLoc VARCHAR2,
                     schemaDoc XMLType,
                     valid     IN PLS_INTEGER DEFAULT NULL,
                     auxLoc    VARCHAR2 DEFAULT NULL);
```

Table 3–18 DBMS_ORAXBRL.LOADSCHEMA Parameters

Parameter	Description
schemaLoc	Location in the XBRL repository of schema document schemaDoc. This must be an absolute URI, by which other documents can refer to this taxonomy schema (using element schemaRef). Raise an error if NULL. Any relative URIs used as the values of attributes xlink:href and schemaLocation in document schemaDoc are interpreted relative to location schemaLoc. You must ensure that schemaLoc is the proper base location for any such relative URIs.
schemaDoc	XBRL schema document. Raise an error if NULL.
valid	Validity of document schemaDoc. 1: valid, 0: invalid.
auxLoc	XBRL repository location of an XBRL auxiliary document that is mapped to schemaDoc.

mapPublishedLocation

Map an Oracle XML DB Repository path to an HTTP URL. This applies a published location (a URL) recursively to all files and folders under the specified Oracle XML DB Repository folder.

Each repository path starts with /XBRL/ followed by a user name. These top two levels of the path are, in effect, replaced by the URL that you provide as the published location.

For example, if a document is loaded into the XBRL repository at path /XBRL/some-user/us-gaap/1.0/elts/us-gaap-std-2008-03-31.xsd, and you invoke mapPublishedLocation('/XBRL/some-user/us-gaap', 'http://xbrl.us'), then the file is published at http://xbrl.us/us-gaap/1.0/elts/us-gaap-std-2008-03-31.xsd.

You can optionally exclude all repository documents below a given level from being published. They are then ignored for XBRL purposes; in particular, they are not available for discovery. You do this be specifying the optional parameter levels. Documents at a depth greater than levels below folderpath are ignored.

Invoke this procedure after you use protocols to upload XBRL documents.

```
PROCEDURE mapPublishedLocation (folderpath VARCHAR2,
                                publishedLocation VARCHAR2,
                                levels PLS_INTEGER);
```

Table 3–19 DBMS_ORAXBRL.MAPPUBLISHEDLOCATION Parameters

Parameter	Description
folderpath	XBRL repository folder to be mapped to publishedLocation.
publishedLocation	URL to be mapped to folderpath.
levels	Number of levels below folderpath for which to make documents available for discovery. By default there is no limit: all documents below folderpath are made available for discovery.

registerTaxonomySchema

Register one taxonomy schema with the XBRL repository. Invoke this after calling loadSchema. Needed for any schema that has tuple elements.

```
PROCEDURE registerTaxonomySchema(schemaLoc  VARCHAR2,
                                 annotation XMLType DEFAULT NULL,
```

```
                                force      IN PLS_INTEGER DEFAULT 0);
```

Note: Before invoking `registerTaxonomySchema` for the first time, you must run the script `xbrlregschema.sql` to register the standard XBRL schemas. See "XBRLScripts" on page 5-6.

Table 3–20 DBMS_ORXBRL.REGISTERTAXONOMYSCHEMA Parameters

Parameter	Description
`schemaLoc`	Location in the XBRL repository of the XBRL schema to be registered. This must be an absolute URI, by which other documents can refer to this taxonomy schema. Raise an error if `NULL`.
`annotation`	List of XBRL tuple elements, with their names and default table names. During registration, create the table with the specified default name for storing the XBRL tuple data. Do not create the table if `NULL` (the default value).
`force`	`1` or greater: Do not raise an error if the schema is invalid. You can use `1` when registering a set of schemas with circular dependencies, to avoid raising errors due to those dependencies. `0` or less: Raise an error if the schema to be registered is invalid.

Example 3–1 shows the XML format for parameter `annotation`.

Example 3–1 Format of Parameter annotation

```
<schemaAnnotation>
  <element>
    <elementName>. . .</elementName>
    <defaultTableName>. . .</defaultTableName>
  </element>
</schemaAnnotation>
```

updateDocValidity

Update the validity status of a document and associate an auxiliary document with it. Use this procedure after validation. (To update the auxiliary document, use procedure `loadAuxDocument`.)

```
PROCEDURE updateDocValidity(xbrlLoc    VARCHAR2,
                            valid      PLS_INTEGER DEFAULT NULL,
                            auxDocPath VARCHAR2 DEFAULT NULL);
```

Table 3–21 DBMS_ORAXBRL.UPDATEDOCVALIDITY Parameters

Parameter	Description
`xbrlLoc`	Location in the XBRL repository of the document whose validity is being updated.
`valid`	New validation status of the document at location `xbrlLoc`. `1` means valid; `0` means invalid.
`auxDocPath`	Location in the XBRL repository of the auxiliary document to associate with the document at location `xbrlLoc`.

validateDTSIntegrity, validateDTSIntegrity2

Check whether all referenced taxonomy schemas and linkbases exist in the XBRL repository. Return the list of taxonomies as an XML document, indicating which

taxonomies are missing. Reflects the state the XBRL repository at the time the procedure is invoked.

```
FUNCTION validateDTSIntegrity(entryURI VARCHAR2) RETURN XMLType;
FUNCTION validateDTSIntegrity2(entryURI VARCHAR2) RETURN ORAXBRL_CONCEPTLIST;
```

Table 3–22 DBMS_ORXBRL.VALIDATEDTSINTEGRITY(2) Parameters

Parameter	Description
`entryURI`	The entry URI into a taxonomy. Raise an error if `NULL`.

Return an XML document that contains the locations of the documents in the discoverable taxonomy set (DTS) and an indication of whether that XML document is in XBRL repository. Example 3–2 illustrates this.

Example 3–2 validateDTSIntegrity

```
SELECT DBMS_ORAXBRL.validateDTSIntegrity(' http://xbrl.boa.com/2006-12-31')
  FROM DUAL;

<DiscoverSet>
  <DocumentPath InRepository="True">boa-20061231.xsd< /DocumentPath>
  <DocumentPath InRepository="True">
http://xbrl.us/us-gaap/1.0/elts/us-gaap-std-2008-03-31.xsd</DocumentPath>
  <DocumentPath InRepository="True" >
http://xbrl.us/us-gaap/1.0/elts/us-types-2008-03-31.xsd</DocumentPath>
  <DocumentPath InRepository="True" >
http://xbrl.us/us-gaap/1.0/elts/us-types-2008-03-31.xsd</DocumentPath>
  <DocumentPath InRepository="False" > boa-20061231_pre.xml</DocumentPath>
  <DocumentPath InRepository="True" > boa-20061231_cal.xml</DocumentPath>
. . .
</DiscoverSet>
```

XBRL Repository Query

You can query the XBRL content in an XBRL repository directly, using the XQuery language. XBRL Extension to Oracle XML DB also provides the following:

- Relational (third normal form) views over XBRL content, for relational queryability – see "XBRL Relational Representation"
- Network generation APIs, to reconstruct XBRL networks – see "Instance Network Functions: DBMS_ORAXBRLI" and "Instance Network Functions: DBMS_ORAXBRLI"
- Transforming procedures, to construct derived views – see "Transforming Procedures: DBMS_ORAXBRLV"

XBRL Relational Representation

XBRL Extension to Oracle XML DB provides relational views of your XBRL content. This XBRL relational representation is a third normal form data model. It gives you simple access to attributes of schemas, linkbases, targets of linkbases, and items in an instance document. You can query these views directly, or you can create derived views over them to extract a particular representation (see steps 3 and 5, "Building and Using a Sample XBRL Application: USGAAP 2008" on page 2-4).

Table 3–23 Relational Views of XBRL Content

Type	Name	Description
Schema	ORAXBRL_XS_TARGETNSV	Target namespace used in an XBRL schema document.
	ORAXBRL_XS_NSV	Namespaces referenced by an XBRL schema document.
	ORAXBRL_XS_IMPORTNSV	Schemas imported by an XBRL schema document.
	ORAXBRL_XS_LINKBASEREFV	Linkbases referenced by an XBRL schema document
	ORAXBRL_XS_ROLETYPEV	`roletype` elements used in an XBRL taxonomy.
	ORAXBRL_XS_ARCROLETYPEV	`arcroletype` elements used in an XBRL taxonomy.
	ORAXBRL_XS_ELEMENT	Elements used in an XBRL taxonomy.
	ORAXBRL_XS_GROUPV	Element groups used in and XBRL taxonomy.
	ORAXBRL_XS_COMPLEXTYPESC	Complex types defined in an XBRL taxonomy.
Linkbase	ORAXBRL_PRES_LINKBASE	Presentation arcs defined in an XBRL taxonomy.
	ORAXBRL_CALCULATION_LINKBASE	Calculation arcs defined in an XBRL taxonomy.
	ORAXBRL_DEFINITION_LINKBASE	Definition arcs defined in an XBRL taxonomy.
	ORAXBRL_LABEL_LINKBASE	Label arcs defined in an XBRL taxonomy.
	ORAXBRL_REFERENCE_LINKBASE	Reference arcs defined in an XBRL taxonomy.
Instance	ORAXBRL_INST_SCHEMAREFV	XBRL schemas referenced by an XBRL instance document.
	ORAXBRL_INST_LINKBASEREFV	Linkbases referenced by anXBRL instance document.
	ORAXBRL_INST_ROLEREFV	Role references from an XBRL instance document.
	ORAXBRL_INST_ARCROLEREFV	`arcroleref` elements from an XBRL instance document.
	ORAXBRL_INST_NSV	Namespaces used in an XBRL instance document.
	ORAXBRL_INST_UNITV	Unit definitions in an XBRL instance document.
	ORAXBRL_INST_CONTEXTV	Context definitions in an XBRL instance document.
	ORAXBRL_FOOTNOTES	Footnotes defined in an XBRL instance document.
	ORAXBRL_SEGMENT_EXPLICITV	Explicit dimensional attributes defined in the segment part of an XBRL instance document.
	ORAXBRL_SCENARIO_EXPLICITV	Explicit dimensional attributes defined in the scenario part of an XBRL instance document.

Table 3–23 (Cont.) Relational Views of XBRL Content

Type	Name	Description
	ORAXBRL_SEGMENT_TYPEDV	Typed dimensional attributes defined in the segment part of an XBRL instance document.
	ORAXBRL_SCENARIO_TYPEDV	Typed dimensional attributes defined in the scenario part of an XBRL instance document.
	ORAXBRL_INST_ITEMV	Fact values reported in an XBRL instance document.

Instance Network Functions: DBMS_ORAXBRLI

The instance network functions are part of PL/SQL package DBMS_ORAXBRLI. You can use these functions to generate XBRL reports that combine taxonomy and instance data.

There are essentially two versions of each function. The version whose name ends in "2" returns an instance of data type ORAXBRL_CONCEPTLIST (or ORAXBRL_ITEMLIST, in the case of multiple_instance_network2). The version whose name has no "2" appended to it returns an XMLType instance.

Table 3–24 DBMS_ORAXBRI Repository Query APIs

Function	Description
instance_network, instance_network2	Return reported data, organized by a base set of concept-concept relationships, such as a presentation tree.
multiple_instance_network, multiple_instance_network2	Return reported data across multiple instance documents, organized by a base set of concept-concept relationships, such as a presentation tree.

instance_network, instance_network2

Return reported data, organized by a base set of concept-concept relationships, such as a presentation tree.

```
FUNCTION instance_network (entryURI             VARCHAR2,
                           entity               VARCHAR2,
                           periodstart          DATE,
                           periodend            DATE,
                           network_eLinkRoleURI VARCHAR2,
                           network_arcNSURI     VARCHAR2,
                           network_arcLocalName VARCHAR2,
                           network_arcRoleURI   VARCHAR2,
                           label_eLinkRoleURI   VARCHAR2,
                           label_arcRoleURI     VARCHAR2,
                           label_roleURI        VARCHAR2,
                           lang                 VARCHAR2,
                           includeReference     PLS_INTEGER)
   RETURN XMLType;[1]
```

[1] Function instance_network2 has the same signature, except it returns an instance of data type ORAXBRL_CONCEPTLIST.

Table 3–25 DBMS_ORAXBRI.INSTANCE_NETWORK(2) Parameters

Parameter	Description
entryURI	Entry URI into the taxonomy. Raise an error if NULL.
entity	Entity identifier. Raise an error if NULL.
periodStart	Starting period. Raise an error if NULL.
periodEnd	Ending period. Raise an error if NULL.
network_eLinkRoleURI	Base set extended link role (xlink:role). Default (NULL): http://www.xbrl.org/2003/role/link
network_arcNSURI	Base set namespace URI. Default (NULL): http://www.xbrl.org/2003/linkbase
network_arcLocalName	Base set local name. Default (NULL): presentationArc
network_arcRoleURI	Base set resource arc role (xlink:arcrole). Default (NULL): http://www.xbrl.org/2003/arcrole/parent-child
label_eLinkRoleURI	Label extended link role (xlink:role). Default (NULL): http://www.xbrl.org/2003/role/link
label_arcRoleURI	Label resource arc role (xlink:arcrole). Default (NULL): http://www.xbrl.org/2003/arcrole/concept-label
label_roleURI	Label role (xlink:role). Default (NULL): http://www.xbrl.org/2003/role/label
lang	Language (xml:lang). Default (NULL): en
includeReference	*Ignored.*

multiple_instance_network, multiple_instance_network2

Return reported data across multiple instance documents, organized by a base set of concept-concept relationships, such as a presentation tree.

```
FUNCTION multiple_instance_network(entryURI             VARCHAR2,
                                   entityList           VARCHAR2,
                                   periodstart          DATE,
                                   periodend            DATE,
                                   network_eLinkRoleURI VARCHAR2,
                                   network_arcNSURI     VARCHAR2,
                                   network_arcLocalName VARCHAR2,
                                   network_arcRoleURI   VARCHAR2,
                                   label_eLinkRoleURI   VARCHAR2,
                                   label_arcRoleURI     VARCHAR2,
                                   label_roleURI        VARCHAR2,
                                   lang                 VARCHAR2,
                                   includeReference     PLS_INTEGER)
  RETURN XMLType;[2]
```

[2] Function multiple_instance_network2 has the same signature, except it returns an instance of data type ORAXBRL_ITEMLIST.

Table 3–26 DBMS_ORAXBRI.MULTIPLE_INSTANCE_NETWORK(2) Parameters

Parameter	Description
`entryURI`	Entry URI into the taxonomy. Raise an error if `NULL`.
`entityList`	A comma-delimited list of entities: *`entity1`*,*`entity2`*,... *`entityN`*. Raise an error if `NULL`.
`periodStart`	The starting period. Raise an error if `NULL`.
`periodEnd`	The ending period. Raise an error if `NULL`.
`network_eLinkRoleURI`	Base set extended link role (`xlink:role`). Default (`NULL`): `http://www.xbrl.org/2003/role/link`
`network_arcNSURI`	Base set namespace URI. Default (`NULL`): `http://www.xbrl.org/2003/linkbase`
`network_arcLocalName`	Base set local name. Default (`NULL`): `presentationArc`
`network_arcRoleURI`	Base set resource arc role (`xlink:arcrole`). Default (`NULL`): `http://www.xbrl.org/2003/arcrole/parent-child`
`label_eLinkRoleURI`	Label extended link role (`xlink:role`). Default (`NULL`): `http://www.xbrl.org/2003/role/link`
`label_arcRoleURI`	Label resource arc role (`xlink:arcrole`). Default (`NULL`): `http://www.xbrl.org/2003/arcrole/concept-label`
`label_roleURI`	Label role (`xlink:role`). Default (`NULL`): `http://www.xbrl.org/2003/role/label`
`lang`	Language (`xml:lang`). Default (`NULL`): `en`
`includeReference`	*Ignored.*

Concept Network Functions: DBMS_ORAXBRLT

The concept network functions are part of PL/SQL package `DBMS_ORAXBRLT`. They generate XBRL taxonomy hierarchies. There are essentially two versions of each function. The version whose name ends in "2" returns an instance of data type `ORAXBRL_CONCEPTLIST`. The version whose name has no "2" appended to it returns an `XMLType` instance.

Table 3–27 DBMS_ORAXBRT Concept Network Function APIs

Function	Description
`concept_roots, concept_roots2`	Return the root nodes that correspond to a given DTS, entry URI, and XLink role.
`concepts_in_tree, concepts_in_tree2`	Return the concepts that are the descendents of a particular node in a base set tree, with labels. If no entry URI is specified, then return all concepts with the specified XLink role.
`concepts_network, concepts_network2`	Return a view of a base set of concept-concept relationships, such as a presentation tree.

concepts_network, concepts_network2

Return a view of a base set of concept-concept relationships, such as a presentation tree.

```
FUNCTION concepts_network(entryURI             VARCHAR2,
                          network_linkRoleURI  VARCHAR2,
                          network_arcNSURI     VARCHAR2,
                          network_arcLocalName VARCHAR2,
                          network_arcRoleURI   VARCHAR2,
                          labeleLinkRoleURI    VARCHAR2,
                          labelarcRoleURI      VARCHAR2,
                          label_roleURI        VARCHAR2,
                          lang                 VARCHAR2,
                          includeReference     PLS_INTEGER)
  RETURN XMLType;[3]
```

Table 3–28 DBMS_ORAXBRT.CONCEPTS_NETWORK(2) Parameters

Parameter	Description
`entryURI`	Entry URI into a taxonomy. Raise an error if `NULL`.
`network_eLinkRoleURI`	Base set extended link role (`xlink:role`). Default (`NULL`): `http://www.xbrl.org/2003/role/link`
`network_arcNSURI`	Base set namespace URI. Default (`NULL`): `http://www.xbrl.org/2003/linkbase`
`network_arcLocalName`	Base set local name. Default (`NULL`): `presentationArc`
`network_arcRoleURI`	Base set resource arc role (`xlink:arcrole`). Default (`NULL`): `http://www.xbrl.org/2003/arcrole/parent-child`
`label_eLinkRoleURI`	Label extended link role (`xlink:role`). Default (`NULL`): `http://www.xbrl.org/2003/role/link`
`label_arcRoleURI`	Label resource arc role (`xlink:arcrole`). Default (`NULL`): `http://www.xbrl.org/2003/arcrole/concept-label`
`label_roleURI`	Label role (`xlink:role`). Default (`NULL`): `http://www.xbrl.org/2003/role/label`
`lang`	Language (`xml:lang`). Default (`NULL`): `en`
`includeReference`	*Ignored.*

concept_roots, concept_roots2

Return the root nodes that correspond to a given DTS, entry URI, and XLink role.

```
FUNCTION concept_roots(entryURI             VARCHAR2,
                       network_eLinkRoleURI VARCHAR2(200),
                       network_arcNSURI     VARCHAR2(200),
                       network_arcLocalName VARCHAR2(200),
```

[3] Function `concepts_network2` has the same signature, except it returns an instance of data type `ORAXBRL_CONCEPTLIST`.

```
                        network_arcRoleURI   VARCHAR2(200),
                        label_eLinkRoleURI   VARCHAR2(200),
                        label_arcRoleURI     VARCHAR2(200),
                        label_roleURI        VARCHAR2(200),
                        lang                 VARCHAR2(200),
                        includeReference     PLS_INTEGER)
  RETURN XMLType;
```
[4]

Table 3–29 DBMS_ORAXBRT.CONCEPT_ROOTS(2) Parameters

Parameter	Description
`entryURI`	Entry URI for a taxonomy. Raise an error if `NULL`.
`network_eLinkRoleURI`	Base set extended link role (`xlink:role`). Default (`NULL`): `http://www.xbrl.org/2003/role/link`
`network_arcNSURI`	Base set namespace URI. Default (`NULL`): `http://www.xbrl.org/2003/linkbase`
`network_arcLocalName`	Base set local name. Default (`NULL`): `presentationArc`
`network_arcRoleURI`	Base set resource arc role (`xlink:arcrole`). Default (`NULL`): `http://www.xbrl.org/2003/arcrole/parent-child`
`label_eLinkRoleURI`	Label extended link role (`xlink:role`). Default (`NULL`): `http://www.xbrl.org/2003/role/link`
`label_arcRoleURI`	Label resource arc role (`xlink:arcrole`). Default (`NULL`): `http://www.xbrl.org/2003/arcrole/concept-label`
`label_roleURI`	Label role (`xlink:role`). Default (`NULL`): `http://www.xbrl.org/2003/role/label`
`lang`	Language (`xml:lang`). Default (`NULL`): `en`
`includeReference`	*Ignored.*

concepts_in_tree, concepts_in_tree2

Return the concepts that are descendents of a particular node in a base set tree, with labels. If no entry URI is specified, then return all concepts.

```
FUNCTION concepts_in_tree(entry                VARCHAR2,
                          concept_namespaceURI VARCHAR2,
                          concept_name         VARCHAR2,
                          network_eLinkRoleURI VARCHAR2,
                          network_arcNSURI     VARCHAR2,
                          network_arcLocalName VARCHAR2,
                          network_arcRoleURI   VARCHAR2,
                          label_eLinkRoleURI   VARCHAR2,
                          label_arcRoleURI     VARCHAR2,
                          label_roleURI        VARCHAR2,
                          lang                 VARCHAR2,
                          includeReference     PLS_INTEGER)
```

[4] Function `concept_roots2` has the same signature, except it returns an instance of data type `ORAXBRL_CONCEPTLIST`.

```
RETURN XMLTYPE;
```
[5]

Table 3–30 DBMS_ORAXBRT.CONCEPTS_IN_TREE(2) Parameters

Parameter	Description
entry	Entry into a taxonomy. Raise an error if NULL.
concept_namespaceURI	Concept namespace URI. Raise an error if NULL.
concept_name	Concept name. Raise an error if NULL.
network_eLinkRoleURI	Base set extended link role (xlink:role). Default (NULL): http://www.xbrl.org/2003/role/link
network_arcNSURI	Base set namespace URI. Default (NULL): http://www.xbrl.org/2003/linkbase
network_arcLocalName	Base set local name. Default (NULL): presentationArc
network_arcRoleURI	Base set resource arc role (xlink:arcrole). Default (NULL): http://www.xbrl.org/2003/arcrole/parent-child
label_eLinkRoleURI	Label extended link role (xlink:role). Default (NULL): http://www.xbrl.org/2003/role/link
label_arcRoleURI	Label resource arc role (xlink:arcrole). Default (NULL): http://www.xbrl.org/2003/arcrole/concept-label
label_roleURI	Label role (xlink:role). Default (NULL): http://www.xbrl.org/2003/role/label
lang	Language (xml:lang). Default (NULL): en
includeReference	*Ignored.*

Transforming Procedures: DBMS_ORAXBRLV

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

- The XBRL relational representation
- The network generation APIs
- Dimensional information

The transforming procedures are in PL/SQL package DBMS_ORAXBRLV.

Table 3–31 DBMS_ORAXBRV Transforming Procedure APIs

Procedure	Description
createFactTable (Deprecated)	Create a fact table as a view.

[5] Function concepts_in_tree2 has the same signature, except it returns an instance of data type ORAXBRL_CONCEPTLIST.

Table 3–31 (Cont.) DBMS_ORAXBRV Transforming Procedure APIs

Procedure	Description
`createHyperCubeFactTable`	Search the hypercube network of the given primary item to find valid dimensions. Create a star schema, which is a fact table plus dimension tables. Optionally create a join view between the fact table and the dimension tables.
`createHyperCubeSuperFactTable`	Search for primary items that include the given hypercube. Create a super fact table and dimension tables. Optionally create a join view between the super fact table and the dimension tables.
`createStarSchemaFromFact`	Search the hypercube network of the given primary item to find valid dimensions. Create a star schema, which is a fact table plus dimension tables. Optionally create a join view between the fact table and the dimension tables. Optionally cache the names of the created tables.
`createStarSchemaFromHC`	Search for primary items that include the given hypercube. Create a super fact table and dimension tables. Optionally create a join view between the super fact table and the dimension tables. Optionally cache the names of the created tables.
`createSuperFactTable (Deprecated)`	Create a super fact table as a view.
`createViewForConceptRoots`	Create a relational view for PL/SQL function `concepts_roots`.
`createViewForConceptTree`	Create a relational view for PL/SQL functions `concepts_network` and `concepts_in_tree`.
`createViewForInstanceNetwork`	Create a relational view for PL/SQL functions `instance_network` and `multiple_instance_network`.
`dropStarSchema`	Drop a star schema: the fact table or super fact table, dimension tables, and join view (if any) that are identified by a given `tableName`, provided such information is cached in a system table.

createFactTable (Deprecated)

Create a fact table as a view.

> **Note:** Procedure `createFactTable` is deprecated as of XBRL Extension to Oracle XML DB 11*g* Release 2 (11.2.0.2.1).

```
PROCEDURE createFactTable(tableName           VARCHAR2,
                          entity              VARCHAR2,
                          entryURI            VARCHAR2,
                          conceptNamespaceURI VARCHAR2,
                          conceptLocalName    VARCHAR2);
```

Table 3–32 DBMS_ORAXBRV.CREATEFACTTABLE Parameters (Deprecated)

Parameter	Description
`tableName`	Name of the fact-table view. Raise an error if `NULL`.
`entity`	Entity identifier. Raise an error if `NULL`.

Table 3–32 (Cont.) DBMS_ORAXBRV.CREATEFACTTABLE Parameters (Deprecated)

Parameter	Description
entryURI	Entry URI into the taxonomy. Raise an error if NULL. It must be the target namespace of the schema specified in element schemaRef of the instance document. (entity, entryURI) uniquely identifies an XBRL instance document.
conceptNamespaceURI	URI for the concept namespace. Raise an error if NULL.
conceptLocalName	Local name of the concept. Raise an error if NULL.

Example 3–3 createFactTable

```
createFactTable('SALES', 'SAMP', 'http://www.SampleCompany.com/Company',
                'http://www.example.com/Patterns/Sales', 'Sales');
```

This creates the fact-table view SALES. See Example 3–5.

createHyperCubeFactTable

Search the hypercube network of a given primary item to find valid dimensions. Create a star schema, which is a fact table plus dimension tables. If argument tableName is not NULL then also create a join view between the fact table and the dimension tables. The pair (entity, entryURI) uniquely identifies an XBRL instance document.

```
PROCEDURE createHyperCubeFactTable(tableName           VARCHAR2,
                                   entity              VARCHAR2,
                                   entryURI            VARCHAR2,
                                   conceptNamespaceURI VARCHAR2,
                                   conceptLocalName    VARCHAR2,
                                   xlinkRole           VARCHAR2);
```

For a fact[6] name to be usable as parameter conceptLocalName, the fact should have associated dimension information. Search instance documents for occurrences of a fact you are interested in. If you find that the fact is associated with dimension information then it is a candidate for use with procedure createHyperCubeFactTable.

Table 3–33 DBMS_ORAXBRV.CREATEHYPERCUBEFACTTABLE Parameters

Parameter	Description
tableName	If NULL, create a fact table and dimension tables (only). Otherwise, create a fact table, dimension tables, and a join (as a view named tableName) between the fact table and the dimension tables.
entity	Entity identifier. Raise an error if NULL.
entryURI	Entry URI into a taxonomy. The target namespace of the schema that is specified in element schemaRef of the instance document. Raise an error if NULL.
conceptNamespaceURI	Concept namespace URI. Raise an error if NULL.
conceptLocalName	Concept local name. Raise an error if NULL.

[6] A **fact** in an instance document corresponds to a concept in a taxonomy.

Table 3–33 (Cont.) DBMS_ORAXBRV.CREATEHYPERCUBEFACTTABLE Parameters

Parameter	Description
xLinkRole	Extended link role (xlink:role) of the base sets that contain the has-hypercube arcs, that is, the arcs that have arc role http://xbrl.org/int/dim/arcrole/all or http://xbrl.org/int/dim/arcrole/notAll. Raise an error if NULL.

`createHyperCubeFactTable(t, e, u, cnu, cln, xr)` is equivalent to `createStarSchemaFromFact(t, e, u, cnu, cln, xr, 1, 0)`, if `t` is not `NULL`.

`createHyperCubeFactTable(NULL, e, u, cnu, cln, xr)` is equivalent to `createStarSchemaFromFact('DUMMY', e, u, cnu, cln, xr, 0, 0)`.

Example 3–4 createHyperCubeFactTable

```
createHyperCubeFactTable('SALES_DIM',
                         'SAMP',
                         'http://www.SampleCompany.com/Company',
                         'http://www.example.com/Patterns/Sales',
                         'Sales',
                         'http://www.SampleCompany.com/PrimaryConcepts');
```

This creates fact table *user*_SALES, dimension tables *user*_BYPRODUCTPLACEHOLDER and *user*_BYREGIONPLACEHOLDER, and view SALES_DIM, where *user* is the database user logged in when `createHyperCubeFactTable` is invoked. These tables are shown in Example 3–5, Example 3–6, and Example 3–7.

Example 3–5 Fact Table user_SALES

```
Name                                      Null?    Type
----------------------------------------- -------- ---------------------
CONTEXT_ID                                         VARCHAR2(2048)
COMPANY_NAME                                       VARCHAR2(4000)
START_DATE                                         DATE
END_DATE                                           DATE
INSTANT_DATE                                       DATE
VALUE                                              CLOB
```

Example 3–6 Dimension Tables user_BYPRODUCTPLACEHOLDER and user_BYREGIONPLACEHOLDER

```
Name                                      Null?    Type
----------------------------------------- -------- ---------------------
CONTEXT_ID                                         VARCHAR2(4000)
DOMAIN_VALUE                                       VARCHAR2(4000)
```

Column CONTEXT_ID is the primary key for the dimension table, and column DOMAIN_VALUE contains the value of the dimension domain members.

Example 3–7 Join View SALES_DIM

```
Name                                      Null?    Type
----------------------------------------- -------- ---------------------
COMPANY_NAME                                       VARCHAR2(4000)
START_DATE                                         DATE
END_DATE                                           DATE
INSTANT_DATE                                       DATE
BYREGIONPLACEHOLDER                                VARCHAR2(4000)
```

```
BYPRODUCTPLACEHOLDER                                VARCHAR2(4000)
VALUE                                               CLOB
```

createHyperCubeSuperFactTable

Search for primary items that include the given hypercube. Create a super fact table and dimension tables. If argument `tableName` is not `NULL` then also create a join view between the super fact table and the dimension tables.

A **super fact table** can contain more than one kind of fact. It acts like a collection of fact tables that each contain one kind of fact. It contains all of the primary items associated with a given hyper cube.

```
PROCEDURE createHyperCubeSuperFactTable(tableName         VARCHAR2,
                                        entity            VARCHAR2,
                                        entryURI          VARCHAR2,
                                        HCNamespaceURI    VARCHAR2,
                                        HCLocalName       VARCHAR2,
                                        HCXlinkRole       VARCHAR2,
                                        HCContextElement  VARCHAR2,
                                        HCTargetRole      VARCHAR2);
```

Table 3–34 DBMS_ORAXBRV.CREATEHYPERCUBESUPERFACTTABLE Parameters

Parameter	Description
`tableName`	If `NULL`, create a fact table and dimension tables (only). Otherwise, create a fact table, dimension tables, and a join (as a view named `tableName`) between the fact table and the dimension tables.
`entity`	Entity identifier. Raise an error if `NULL`.
`entryURI`	Entry URI into the taxonomy. Raise an error if `NULL`. It must be the target namespace of the schema specified in element `schemaRef` of the instance document. (`entity`, `entryURI`) uniquely identifies an XBRL instance document.
`HCNamespaceURI`	Namespace URI of the hypercube. Raise an error if `NULL`.
`HCLocalName`	Local name of the hypercube. Raise an error if `NULL`.
`HCXLinkRole`	XLink role (`xlink:role`) of the base sets that contain arc `has-hypercube`. Raise an error if `NULL`.
`HCContextElement`	Value of attribute `contextElement` specified in arc `has-hypercube`. Raise an error if `NULL`.
`HCTargetRole`	Value of attribute `targetRole` specified in arc `has-hypercube`. Raise an error if `NULL`.

`createHyperCubeSuperFactTable(t, e, u, nu, ln, xr, ce, tr)` is equivalent to `createStarSchemaFromHC(t, e, u, nu, ln, xr, ce, tr, 1, 0)`, if `t` is not `NULL`.

`createHyperCubeSuperFactTable(NULL, e, u, nu, ln, xr, ce tr)` is equivalent to `createStarSchemaFromHC('DUMMY', e, u, nu, ln, xr, ce, tr, 0, 0)`.

Example 3–8 createHyperCubeSuperFactTable

```
createHyperCubeSuperFactTable(
   'BOA_STATEMENT',
   '0000070858',
   'http://xbrl.boa.com/2006-12-31',
   'http://xbrl.us/us-gaap/2008-03-31',
   'StatementTable',
```

```
  'http://xbrl.boa.com/2006-12-31/ext/StockholdersEquity',
  'segment', 'http://xbrl.boa.com/2006-12-31/ext/StockholdersEquity');
```

This creates fact table *user*_STATEMENTTABLE, dimension table *user*_EQUITYCOMPONENTSAXIS, and join view BOA_STATEMENT, where *user* is the database user logged in when createHyperCubeSuperFactTable is invoked. These tables and view are shown in Example 3–9, Example 3–10, and Example 3–11.

Example 3–9 Fact Table user_STATEMENTTABLE

```
Name                                      Null?    Type
----------------------------------------- -------- ---------------------
ITEM_NAME                                          VARCHAR2(4000)
CONTEXT_ID                                         VARCHAR2(2048)
COMPANY_NAME                                       VARCHAR2(4000)
START_DATE                                         DATE
END_DATE                                           DATE
INSTANT_DATE                                       DATE
VALUE                                              CLOB
```

Example 3–10 Join View BOA_STATEMENT

```
Name                                      Null?    Type
----------------------------------------- -------- ---------------------
ITEM_NAME                                          VARCHAR2(4000)
COMPANY_NAME                                       VARCHAR2(4000)
START_DATE                                         DATE
END_DATE                                           DATE
INSTANT_DATE                                       DATE
EQUITYCOMPONENTSAXIS                               VARCHAR2(4000)
VALUE                                              CLOB
```

Example 3–11 Dimension Table user_EQUITYCOMPONENTSAXIS

```
Name                                      Null?    Type
----------------------------------------- -------- ---------------------
ITEM_NAME                                          VARCHAR2(4000)
CONTEXT_ID                                         VARCHAR2(2048)
COMPANY_NAME                                       VARCHAR2(4000)
START_DATE                                         DATE
END_DATE                                           DATE
INSTANT_DATE                                       DATE
VALUE                                              CLOB
```

createStarSchemaFromFact

Search the hypercube network of the primary item to find valid dimensions. Create a star schema, which is a fact table plus dimension tables. Optionally create a join view between the fact table and the dimension tables. Optionally cache the names of the created tables. Return a list of the names of the created tables.

```
FUNCTION createStarSchemaFromFact(tableName VARCHAR2,
                                  entity VARCHAR2,
                                  entryURI VARCHAR2,
                                  conceptNamespaceURI VARCHAR2,
                                  conceptLocalName VARCHAR2,
                                  xlinkRole VARCHAR2,
                                  createJoin PLS_INTEGER DEFAULT 1,
                                  cache PLS_INTEGER DEFAULT 0)
  RETURN ORAXBRL_STARSCHEMA;
```

Together, `entity` plus `entryURI` uniquely identify an XBRL instance document.

Table 3–35 DBMS_ORAXBRV.CREATESTARSCHEMAFROMFACT Parameters

Parameter	Description
`tableName`	Unique identifier for the set comprising the created fact table and dimension tables. If `NULL`, raise an error. If `createJoin` is `1`, `tableName` is also the name of the join view between the fact table and the dimension tables. If `cache` is `1`, cache `tableName` and the names of the fact table and dimension tables in a system table. If neither `createJoin` nor `cache` is `1`, then `tableName` is ignored.
`entity`	Entity identifier. Raise an error if `NULL`.
`entryURI`	Entry URI into a taxonomy. The target namespace of the schema that is specified in element `schemaRef` of the instance document. Raise an error if `NULL`.
`conceptNamespaceURI`	Concept namespace URI. Raise an error if `NULL`.
`conceptLocalName`	Concept local name. Raise an error if `NULL`.
`xLinkRole`	Extended link role (`xlink:role`) of the base sets that contain the `has-hypercube` arcs. Raise an error if `NULL`.
`createJoin`	If and only if `1`, create a join view between the fact table and the dimension tables.
`cache`	If and only if `1`, cache `tableName` and the names of the fact table and dimension tables in a system table.

The fact table created is named **`F`**`_entity_conceptLocalName_####`, where `entity` and `conceptLocalName` are the entity and concept local name inputs, and `####` is a four-digit (decimal) number guaranteed to make the name unique.

The dimension tables created are named **`D`**`_entity_`*`dimensionLocalName`*`_####`, where `entity` is the entity input, *`dimensionLocalName`* is the local name of the valid dimension found in the hypercube network given the primary item, and `####` is a four-digit (decimal) number guaranteed to make the name unique.

Database table names are limited to a maximum of 30 characters. The concept and dimension local names are truncated as needed to ensure this.

createStarSchemaFromHC

Search for primary items that include the given hypercube. Create a super fact table and dimension tables. Optionally create a join view between the super fact table and the dimension tables. Optionally cache the names of the created tables. Return a list of the names of the created tables.

```
FUNCTION createStarSchemaFromHC(tableName VARCHAR2,
                                entity VARCHAR2,
                                entryURI VARCHAR2,
                                HCNamespaceURI VARCHAR2,
                                HCLocalName VARCHAR2,
                                HCxlinkRole VARCHAR2,
                                createJoin PLS_INTEGER DEFAULT 1,
                                cache PLS_INTEGER DEFAULT 0)
  RETURN ORAXBRL_STARSCHEMA;
```

Together, `entity` plus `entryURI` uniquely identify an XBRL instance document.

Table 3–36 DBMS_ORAXBRV.CREATESTARSCHEMAFROMHC Parameters

Parameter	Description
`tableName`	Unique identifier for the set comprising the created super fact table and dimension tables. If `NULL`, raise an error. If `createJoin` is `1`, `tableName` is also the name of the join view between the super fact table and the dimension tables. If `cache` is `1`, cache `tableName` and the names of the fact table and dimension tables in a system table. If neither `createJoin` nor `cache` is `1`, then `tableName` is ignored.
`entity`	Entity identifier. Raise an error if `NULL`.
`entryURI`	Entry URI into a taxonomy. The target namespace of the schema that is specified in element `schemaRef` of the instance document. Raise an error if `NULL`.
`HCNamespaceURI`	Namespace URI of the hypercube. Raise an error if `NULL`.
`HCLocalName`	Local name of the hypercube. Raise an error if `NULL`.
`HCXLinkRole`	XLink role (`xlink:role`) of the base sets that contain arc `has-hypercube`. Raise an error if `NULL`.
`HCContextElement`	Value of attribute `contextElement` specified in arc `has-hypercube`. Raise an error if `NULL`.
`HCTargetRole`	Value of attribute `targetRole` specified in arc `has-hypercube`. Raise an error if `NULL`.
`createJoin`	If and only if `1`, create a join view between the super fact table and the dimension tables.
`cache`	If and only if `1`, cache `tableName` and the names of the super fact table and dimension tables in a system table.

The created fact table and dimension tables are named using the same convention as for createStarSchemaFromFact on page 3-23.

createSuperFactTable (Deprecated)

Create a super fact table as a view. It contains all of the primary items of the specified hypercube.

> **Note:** Procedure `createSuperFactTable` is deprecated as of XBRL Extension to Oracle XML DB 11*g* Release 2 (11.2.0.2.1).

```
PROCEDURE createSuperFactTable(tableName VARCHAR2,
                               entity VARCHAR2,
                               entryURI VARCHAR2,
                               hcNamespaceURI VARCHAR2,
                               hcLocalName VARCHAR2,
                               hcXlinkRole VARCHAR2,
                               hcContextElement VARCHAR2,
                               hcTargetRole VARCHAR2);
```

Table 3–37 DBMS_ORAXBRV.CREATESUPERFACTTABLE Parameters (Deprecated)

Parameter	Description
`tableName`	Name of the super fact-table view. Raise an error if `NULL`.

Table 3–37 (Cont.) DBMS_ORAXBRV.CREATESUPERFACTTABLE Parameters

Parameter	Description
entity	Entity identifier. Raise an error if NULL.
entryURI	Entry URI into the taxonomy. Raise an error if NULL. It must be the target namespace of the schema specified in element schemaRef of the instance document. (entity, entryURI) uniquely identifies an XBRL instance document.
hcNamespaceURI	URI for the hypercube namespace. Raise an error if NULL.
hcLocalName	Local name of the hypercube. Raise an error if NULL.
hcXlinkRole	Extended link role (xlink:role) of the base sets that contain arc has-hypercube.
hcContextElement	Value of attribute contextElement that is specified in arc has-hypercube.
hcTargetRole	Value of attribute targetRole that is specified in arc has-hypercube.

Example 3–12 createSuperFactTable

```
createSuperFactTable('STATEMENT', '0000070858', 'http://xbrl.boa.com/2006-12-31',
                     'http://xbrl.us/us-gaap/2008-03-31', 'StatementTable',
                     'http://xbrl.boa.com/2006-12-31/ext/StockholdersEquity',
                     'segment',
                     'http://xbrl.boa.com/2006-12-31/ext/StockholdersEquity');
```

This creates super fact-table view STATEMENT. See Example 3–9.

createViewForConceptRoots

Create a relational view for PL/SQL function concept_roots.

```
PROCEDURE createViewForConceptRoots(viewName             VARCHAR2,
                                    entryURI             VARCHAR2,
                                    network_eLinkRoleURI VARCHAR2,
                                    network_arcNSURI     VARCHAR2,
                                    network_arcLocalName VARCHAR2,
                                    network_arcRoleURI   VARCHAR2,
                                    labeleLinkRoleURI    VARCHAR2,
                                    labelarcRoleURI      VARCHAR2,
                                    label_roleURI        VARCHAR2,
                                    lang                 VARCHAR2);
```

Table 3–38 DBMS_ORAXBRV.CREATEVIEWFORCONCEPTROOTS Parameters

Parameter	Description
viewName	Name of the view. Raise an error if NULL.
entryURI	Entry into a taxonomy. Raise an error if NULL.
network_eLinkRoleURI	Base set extended link role (xlink:role). Default (NULL): http://www.xbrl.org/2003/role/link
network_arcNSURI	Base set namespace URI. Default (NULL): http://www.xbrl.org/2003/linkbase
network_arcLocalName	Base set local name. Default (NULL): presentationArc

Table 3–38 (Cont.) DBMS_ORAXBRV.CREATEVIEWFORCONCEPTROOTS Parameters

Parameter	Description
network_arcRoleURI	Base set resource arc role (xlink:arcrole). Default (NULL): http://www.xbrl.org/2003/arcrole/parent-child
label_eLinkRoleURI	Label extended link role (xlink:role). Default (NULL): http://www.xbrl.org/2003/role/link
label_arcRoleURI	Label resource arc role (xlink:arcrole). Default (NULL): http://www.xbrl.org/2003/arcrole/concept-label
label_roleURI	Label role (xlink:role). Default (NULL): http://www.xbrl.org/2003/role/label
lang	Language (xml:lang). Default (NULL): en

Example 3–13 createViewForConceptRoots

```
createViewForConceptRoots(
  'my_concept_roots',
  'http://xbrl.us/us-gaap-entryPoint-std/2008-03-31',
  'http://xbrl.us/us-gaap/role/statement/StatementOfIncome',
  NULL,
  'presentationArc',
  'http://www.xbrl.org/2003/arcrole/parent-child',
  NULL,
  NULL,
  NULL,
  'en-US');
```

createViewForConceptTree

Create a relational view for PL/SQL functions concepts_network and concepts_in_tree.

```
PROCEDURE createViewForConceptTree(viewName             VARCHAR2,
                                   entryURI             VARCHAR2,
                                   concept_namespaceURI VARCHAR2,
                                   concept_name         VARCHAR2,
                                   network_eLinkRoleURI VARCHAR2,
                                   network_arcNSURI     VARCHAR2,
                                   network_arcLocalName VARCHAR2,
                                   network_arcRoleURI   VARCHAR2,
                                   labeleLinkRoleURI    VARCHAR2,
                                   labelarcRoleURI      VARCHAR2,
                                   label_roleURI        VARCHAR2,
                                   lang                 VARCHAR2,
                                   treeDepth            PLS_INTEGER);
```

Table 3–39 DBMS_ORAXBRV.CREATEVIEWFORCONCEPTTREE Parameters

Parameter	Description
viewName	Name of the view. Raise an error if NULL.
entryURI	Entry into a taxonomy. Raise an error if NULL.
concept_namespaceURI	Concept namespace URI. Raise an error if NULL.

Table 3–39 (Cont.) DBMS_ORAXBRV.CREATEVIEWFORCONCEPTTREE Parameters

Parameter	Description
concept_name	Concept name. By default (NULL), create a relational view for PL/SQL function concepts_network. If not NULL, create a relational view for PL/SQL function concepts_in_tree.
network_eLinkRoleURI	Base set extended link role (xlink:role). Default (NULL): http://www.xbrl.org/2003/role/link
network_arcNSURI	Base set namespace URI. Default (NULL): http://www.xbrl.org/2003/linkbase
network_arcLocalName	Base set local name. Default (NULL): presentationArc
network_arcRoleURI	Base set resource arc role (xlink:arcrole). Default (NULL): http://www.xbrl.org/2003/arcrole/parent-child
label_eLinkRoleURI	Label extended link role (xlink:role). Default (NULL): http://www.xbrl.org/2003/role/link
label_arcRoleURI	Label resource arc role (xlink:arcrole). Default (NULL): http://www.xbrl.org/2003/arcrole/concept-label
label_roleURI	Label Role (xlink:role). Default (NULL): http://www.xbrl.org/2003/role/label
lang	Language (xml:lang). Default (NULL): en
treeDepth	If 1, create a view on the children of the concept. Otherwise, create a view on the descendents of the concept.

Example 3–14 createViewForConceptTree

```
createViewForConceptTree(
  'my_network',
  'http://xbrl.us/us-gaap-entryPoint-std/2008-03-31',
  'http://xbrl.us/us-gaap/2008-03-31',
  'NetIncomeLossAbstract',
  'http://xbrl.us/us-gaap/role/statement/StatementOfIncome',
   NULL,
  'presentationArc',
  'http://www.xbrl.org/2003/arcrole/parent-child',
  NULL,
  NULL,
  NULL,
  'en-US',
  -1);
```

This creates view my_network:

```
Name                                      Null?    Type
----------------------------------------- -------- ---------------------
NAMESPACEURI                                        VARCHAR2(4000)
PREFERREDPREFIX                                     VARCHAR2(4000)
NAME                                                VARCHAR2(4000)
ID                                                  VARCHAR2(4000)
BALANCE                                             VARCHAR2(4000)
```

```
PERIODTYPE                                          VARCHAR2(4000)
ABSTRACT                                            VARCHAR2(4000)
NILLABLE                                            VARCHAR2(4000)
TYPEURI                                             VARCHAR2(4000)
TYPELOCALNAME                                       VARCHAR2(4000)
SGURI                                               VARCHAR2(4000)
SGLOCALNAME                                         VARCHAR2(4000)
ELEM_HREF                                           VARCHAR2(4000)
LABEL                                               VARCHAR2(4000)
PREFERRED_LABEL                                     VARCHAR2(4000)
```

createViewForInstanceNetwork

Create a relational view for PL/SQL functions `instance_network` and `multiple_instance_network`.

```
PROCEDURE createViewForInstanceNetwork(viewName             VARCHAR2,
                                       entryURI             VARCHAR2,
                                       entityList           VARCHAR2,
                                       startDate            DATE,
                                       endDate              DATE,
                                       network_eLinkRoleURI VARCHAR2,
                                       network_arcNSURI     VARCHAR2,
                                       network_arcLocalName VARCHAR2,
                                       network_arcRoleURI   VARCHAR2,
                                       labeleLinkRoleURI    VARCHAR2,
                                       labelarcRoleURI      VARCHAR2,
                                       label_roleURI        VARCHAR2,
                                       lang                 VARCHAR2);
```

Table 3–40 DBMS_ORAXBRV.CREATEVIEWFORINSTANCENETWORK Parameters

Parameter	Description
`viewName`	Name of the view. Raise an error if `NULL`.
`entryURI`	Entry URI to the taxonomy. Raise an error if `NULL`.
`entityList`	A comma-delimited list of entities: *`entity1`*,*`entity2`*,… *`entityN`*. Raise an error if `NULL`. If there is only one entity, it is equivalent to function `instance_network`.
`periodStart`	Starting period. Raise an error if `NULL`.
`periodEnd`	Ending period. Raise an error if `NULL`.
`network_eLinkRoleURI`	Base set extended link role (`xlink:role`). Default (`NULL`): `http://www.xbrl.org/2003/role/link`
`network_arcNSURI`	Base set namespace URI. Default (`NULL`): `http://www.xbrl.org/2003/linkbase`
`network_arcLocalName`	Base set local name. Default (`NULL`): `presentationArc`
`network_arcRoleURI`	Base set resource arc role (`xlink:arcrole`). Default (`NULL`): `http://www.xbrl.org/2003/arcrole/parent-child`
`label_eLinkRoleURI`	Label extended link role (`xlink:role`). Default (`NULL`): `http://www.xbrl.org/2003/role/link`

Table 3–40 (Cont.) DBMS_ORAXBRV.CREATEVIEWFORINSTANCENETWORK

Parameter	Description
`label_arcRoleURI`	Label resource arc role (`xlink:arcrole`). Default (`NULL`): `http://www.xbrl.org/2003/arcrole/concept-label`
`label_roleURI`	Label role (`xlink:role`). Default (`NULL`): `http://www.xbrl.org/2003/role/label`
`lang`	Language (`xml:lang`). Default (`NULL`): `en`

Example 3–15 createViewForInstanceNetwork

```
createViewForInstanceNetwork(
  'my_instance_network',
  'http://xbrl.boa.com/2006-12-31',
  '0000070858',
  1-JAN-05',
    '01-JAN-06',
  'http://xbrl.boa.com/2006-12-31/ext/IncomeStatement',
  NULL,
  'presentationArc',
  'http://www.xbrl.org/2003/arcrole/parent-child',
  NULL,
  NULL,
  NULL,
  'en-US');
```

This creates view `my_instance_network`:

```
Name                                      Null?    Type
----------------------------------------- -------- ---------------------
NAMESPACEURI                                       VARCHAR2(4000)
PREFIX                                             VARCHAR2(4000)
NAME                                               VARCHAR2(4000)
LABEL                                              VARCHAR2(4000)
PREFERRED_LABEL                                    VARCHAR2(4000)
ENTITY_SCHEME                                      VARCHAR2(4000)
ENTITY_IDENTIFIER                                  VARCHAR2(4000)
START_DATE                                         DATE
END_DATE                                           DATE
CONTEXTREF                                         VARCHAR2(4000)
UNITREF                                            VARCHAR2(4000)
DECIMALS                                           VARCHAR2(4000)
VALUE                                              CLOB
```

dropStarSchema

Drop a star schema: the fact table or super fact table, dimension tables, and join view (if any) that are collectively identified by the given `tableName`, provided such information is cached in a system table. (Do nothing if the information is not cached.)

```
PROCEDURE dropStarSchema(tableName VARCHAR2);
```

Table 3–41 DBMS_ORAXBRV.DROPSTARSCHEMA Parameters

Parameter	Description
tableName	tableName argument specified in the invocation of createStarSchemaFromFact or createStarSchemaFromHC that created the star schema to be dropped.

XBRL Extension to Oracle XML DB Data Types

This section describes data types specific to XBRL Extension to Oracle XML DB. They are all in PL/SQL package XBRLSYS.

ORAXBRL_CONCEPT, ORAXBRL_CONCEPTLIST

Data type ORAXBRL_CONCEPTLIST is a varray of ORAXBRL_CONCEPT, which is an object type with the following attributes that pertain to a concept:

Table 3–42 ORAXBRL_CONCEPT Object Type Attributes

Attribute	Type	Description
NAMESPACEURI	VARCHAR2(4000)	Namespace URI of the XML schema that defines the concept.
PREFERREDPREFIX	VARCHAR2(4000)	Preferred prefix for the namespace specified by NAMESPACEURI.
NAME	VARCHAR2(4000)	Local name of the concept.
ID	VARCHAR2(4000)	Unique identifier of the concept.
BALANCE	VARCHAR2(4000)	The credit/debit balance associated with the concept.
PERIODTYPE	VARCHAR2(4000)	Type of the reporting period associated with the concept. Possible values: duration and instant.
ABSTRACT	VARCHAR2(4000)	True means that the concept can be used only in a hierarchy, to group related concepts. Possible values: true and false.
NILLABLE	VARCHAR2(4000)	True means that facts for the concept can be empty. Possible values: true and false.
TYPEURI	VARCHAR2(4000)	Namespace URI of the schema type of the concept.
TYPELOCALNAME	VARCHAR2(4000)	Local name of the schema type of the concept.
SGURI	VARCHAR2(4000)	Namespace URI of the substitution group for the concept.
SGLOCALNAME	VARCHAR2(4000)	Local name of the substitution group for the concept.
ELEM_HREF	VARCHAR2(4000)	Absolute path of the concept in the taxonomy schema.
LABEL	VARCHAR2(4000)	Human-readable name for the concept, unique across the taxonomy.
PREFERRED_LABEL	VARCHAR2(4000)	The preferred label derived from the preferredLabel arc from concept parent to the concept. If the concept has no parent then this is NULL.

ORAXBRL_ITEM, ORAXBRL_ITEMLIST

Data type ORAXBRL_ITEMLIST is a varray of ORAXBRL_ITEM, which is an object type with the following attributes that pertain to a fact:

Table 3–43 ORAXBRL_ITEM Object Type Attributes

Attribute	Type	Description
NAMESPACEURI	VARCHAR2(4000)	Namespace URI of the XML schema that defines the fact.
PREFIX	VARCHAR2(4000)	Prefix for the namespace specified by NAMESPACEURI.
NAME	VARCHAR2(4000)	Local name of the fact.
LABEL	VARCHAR2(4000)	Human-readable name for the fact, unique across the taxonomy.
PREFERRED_LABEL	VARCHAR2(4000)	Same as PREFERRED_LABEL attribute for type ORAXBRL_CONCEPT.
ID	VARCHAR2(4000)	Unique identifier of the fact.
ENTITY_SCHEME	VARCHAR2(4000)	Namespace of the entity identification scheme for the fact.
ENTITY_IDENTIFIER	VARCHAR2(4000)	Value of the entity identifier for the fact.
START_DATE	DATE	Start date for the fact, if the period type is duration. Otherwise, NULL.
END_DATE	DATE	End date for the fact, if the period type is duration. Otherwise, NULL.
CONTEXTREF	VARCHAR2(4000)	A reference to the context associated with the fact.
UNITREF	VARCHAR2(4000)	A reference to the unit associated with the fact.
DECIMALS	VARCHAR2(4000)	Number of decimal places to which numbers have been rounded.
VALUE	VARCHAR2(4000)	Value of the fact.

ORAXBRL_LOCLIST, ORAXBRL_STARSCHEMA

Data type ORAXBRL_LOCLIST is a varray of VARCHAR2(4000). Data type ORAXBRL_STARSCHEMA is an object type with the following attributes that pertain to a star schema:

Table 3–44 ORAXBRL_STARSCHEMA Object Type Attributes

Attribute	Type	Description
FACT_TABLE	VARCHAR2(4000)	Name of the fact table in the star schema.
DIMENSION_LIST	ORAXBRL_LOCLIST	Names of the dimension tables in the star schema.

ORAXBRL_DTSURLLIST, ORAXBRL_DTSURL_T

Data type ORAXBRL_DTSURLLIST is a varray of ORAXBRL_DTSURL_T, which is an object type with the following attributes that pertain to a file in the XBRL repository.

Table 3–45 ORAXBRL_DTSURL_T Object Type Attributes

Attribute	Type	Description
NAME	VARCHAR2(4000)	Name of the file.
TYPE	VARCHAR2(4000)	Type of the file: SCHEMA or LINKBASE.
DOCPATH	VARCHAR2(4000)	Location of the file in the XBRL repository.
XDBREPOPATH	VARCHAR2(4000)	Location of the file in the Oracle XML DB Repository.
OID	RAW(16)	Document or file object ID.

4

Administering XBRL Extension to Oracle XML DB

This chapter covers database administration of XBRL Extension to Oracle XML DB. It includes these topics:

- Overview of XBRL Repository Tables and Indexes
- Creating a Tablespace for the XBRL Repository Indexes
- Partitioning XBRL Repositories
- Obtaining Information about Your XBRL Repository

Note: Refer to "Placeholders in Oracle Database XBRL Extension Developer's Guide" on page x for explanations of the placeholders used here.

Overview of XBRL Repository Tables and Indexes

An XBRL repository uses a set of base tables and a set of automatically generated `XMLIndex` indexes.

The base tables include the following `XMLType` tables with binary XML storage, which store all of your XBRL-related documents. See also Example 4–5.

- `ORA$XBRLSCHEMA` – taxonomy schema documents
- `ORA$XBRLINSTANCE` – instance documents
- `ORA$XBRLLINKBASE` – linkbase documents

There are also other, non-`XMLType` base tables, including these:

- `ORA$XBRLPATH` – has these columns:
  - `doctype`: XBRL document type (`SCHEMA`, `LINKBASE`, or `INSTANCE`)
  - `docpath`: location of the document in the XBRL repository
  - `uploaddate`: time when the document was loaded into the XBRL repository
- `ORA$XBRLNWKCACHE` – a cache for information used during concept generation

`XMLIndex` indexes (with structured components) are created automatically for base tables `ORA$XBRLSCHEMA` and `ORA$XBRLINSTANCE`. These indexes project the values of the structured parts of an XBRL document. They improve performance for queries and analysis of XBRL documents. These indexes have their own storage tables.

See Also:

- Example 4–6, "XMLType Tables and Their XMLIndex Indexes"
- Example 4–7, "XMLIndex Indexes and Their Index Storage Tables"

Creating a Tablespace for the XBRL Repository Indexes

The script that creates an XBRL repository, `xbrlcrt.sql`, takes two parameters that specify the tablespaces to use for the base tables, on the one hand, and the index storage tables, on the other hand. These parameters are *xb_rep_ts* and *xb_rep_idx_ts*, respectively. You can of course use the same value for both parameters if you wish. In that case, the base tables and the index storage tables share the same tablespace.

For an existing XBRL repository, you can use script `xbrlrecidxdrv.sql` to change the tablespace used by the index storage tables. Again, you use parameter *xb_rep_idx_ts* to specify the tablespace to use for this:

```
shell> cd XBRLScripts

SQL+> @xbrlrecidxdrv.sql xb_rep  xb_rep_pass  xb_rep_idx_ts
```

Partitioning XBRL Repositories

For a given XBRL repository, you can partition base table `ORA$XBRLINSTANCE`, which stores your XBRL instance documents. If you do this then the automatically created `XMLIndex` index on that table, `XBRL$INSTANCEIDX`, together with all of its index storage tables, are automatically equipartitioned. Equipartitioning means that there is a corresponding index-table partition for each partition of the base table.

Partioning the XBRL Repository Instance Table

If you want to partition base table `ORA$XBRLINSTANCE` for a given repository then you must edit script `xbrlddl.sql` before you use script `xbrlcrt.sql` to create that repository. In script `xbrlddl.sql`, change the default `CREATE TABLE` statement for table `ORA$XBRLINSTANCE` to one that partitions the table.

Example 4–1 illustrates this. It partitions `XMLType` table `ORA$XBRLINSTANCE` using virtual column `entity_identifier_text`, targeting XML element `identifier` (a child of element `entity` and a grandchild of the first `context` element in an XBRL document).

Example 4–1 Edited CREATE TABLE Statement with Partitioning

```
CREATE TABLE ORA$XBRLINSTANCE of XMLType STORE AS BINARY XML
  VIRTUAL COLUMNS
    (entity_identifier_text AS
      (XMLCast
         (XMLQuery('declare namespace
                      xbrli="http://www.xbrl.org/2003/instance";
                    /xbrli:/xbrl/xbrli:context[1]/xbrli:entity/xbrli:identifier/text()'
                   PASSING OBJECT_VALUE RETURNING CONTENT)
               AS VARCHAR2(50))))
    PARTITION BY RANGE (entity_identifier_text)
      (PARTITION p1 VALUES LESS THAN ('0000066741'),
       PARTITION p2 VALUES LESS THAN ('0000789020')
       PARTITION p3 VALUES LESS THAN (MAXVALUE));
```

You can define such partitioning when you create your XBRL repository after installing XBRL Extension to Oracle XML DB. If you already have an existing repository that you want to partition (or partition differently), you must create a new repository partitioned as needed, and then load it using the data from the previously existing repository. In other words, there is no way to partition an existing repository; you must create a new, partitioned one to replace the existing one.

See Also:

- "Creating an XBRL Repository" on page 5-4, steps 4 and 5 for information about checking the result of executing your updated `CREATE TABLE` statement
- *Oracle XML DB Developer's Guide* for information about partitioning `XMLType` data using virtual columns

Defining Tablespaces for Partitioned XBRL Data

When you create an XBRL repository, any partitions of base table `ORA$XBRLINSTANCE` are in the same tablespace, *`xb_rep_ts`*, and the index storage table partitions are all in the same tablespace, *`xb_rep_idx_ts`*. (See "Creating a Tablespace for the XBRL Repository Indexes" on page 2.)

You can use different tablespaces for different partitions by altering the base table (`ORA$XBRLINSTANCE`) for XBRL instance documents and for the storage tables of the corresponding `XMLIndex` index (`XBRL$INSTANCEIDX`). Example 4–2 and Example 4–3 illustrate this, respectively.

Example 4–2 Changing the Tablespace of Base Table ORA$XBRLINSTANCE

```
ALTER TABLE ORA$XBRLINSTANCE MOVE PARTITION p1 TABLESPACE tbs1
  UPDATE INDEXES (XBRL$INSTANCEIDX(PARTITION p1 PARAMETERS('tablespace tbs1')));
```

Example 4–3 Changing the Tablespace of Index Storage Tables

```
ALTER INDEX XBRL$INSTANCEIDX REBUILD PARTITION p1 PARAMETERS ('tablespace tbs1');
```

Obtaining Information about Your XBRL Repository

This section provides some queries you can use to obtain information about your XBRL data or about the product, XBRL Extension to Oracle XML DB.

Example 4–4 shows how to obtain the version number of the product.

Example 4–4 Version Number for XBRL Extension to Oracle XML DB

```
SELECT value FROM xbrlsys.ora$xbrlrepprop WHERE name='XBRLVERSION';
```

Example 4–5 just lists the `XMLType` tables for the current XBRL repository, that is, `ORA$XBRLSCHEMA`, `ORA$XBRLLINKBASE`, and `ORA$XBRLINSTANCE`.

Example 4–5 XMLType Tables for the XBRL Repository

```
SELECT table_name FROM USER_OBJECT_TABLES;
```

Example 4–6 lists the repository `XMLType` tables and their `XMLIndex` indexes.

Example 4–6 XMLType Tables and Their XMLIndex Indexes

```
SELECT table_name, index_name FROM USER_XML_INDEXES;
```

Example 4–7 lists all of the repository XMLIndex indexes, together with their corresponding index storage tables.

Example 4–7 XMLIndex Indexes and Their Index Storage Tables

```
SELECT DISTINCT p.index_name, t.table_name
  FROM ALL_XML_INDEXES p,
       XMLTable('//xmltab'
                PASSING p.parameters
                COLUMNS table_name VARCHAR2(200) PATH '@name') t
```

Example 4–8 lists the table and tablespace names of the nonpartitioned XMLType tables.

Example 4–8 Nonpartitioned XMLType Tables and Their Tablespaces

```
SELECT table_name, tablespace_name FROM USER_OBJECT_TABLES;
```

Example 4–9 lists the names of all nonpartitioned index storage tables and their tablespaces, for a given XMLIndex index—in this case, index XBRL$INSTANCEIX.

Example 4–9 Nonpartitioned XMLIndex Index Storage Tables and Their Tablespaces

```
SELECT u.table_name, u.tablespace_name FROM USER_TABLES u
  WHERE u.table_name
    IN (SELECT DISTINCT t.table_name
          FROM ALL_XML_INDEXES p,
               XMLTable('//xmltab' PASSING p.parameters
                        COLUMNS table_name VARCHAR2(200) PATH '@name') t
          WHERE index_name = 'XBRL$INSTANCEIDX')
 ORDER BY 1;
```

Example 4–10 obtains the tablespace name for a given nonpartitioned index storage table—in this case, table ORAXBRL_INSTANCE_ITEM.

Example 4–10 Tablespace of a Given Nonpartitioned Index Storage Table

```
SELECT tablespace_name FROM USER_TABLES
  WHERE table_name = 'ORAXBRL_INSTANCE_ITEM';
```

Example 4–11 lists the table names, the index names, and the tablespace name for the nonpartitioned secondary indexes on the index storage tables for a given XMLIndex index—in this case, index XBRL$INSTANCEIX.

Example 4–11 Tables and Tablespace of Secondary Indexes on Index Storage Tables

```
SELECT i.table_name, i.index_name, i.tablespace_name FROM USER_INDEXES i
  WHERE i.table_name
    IN (SELECT DISTINCT t.table_name
          FROM ALL_XML_INDEXES p,
               XMLTable('//xmltab' PASSING p.parameters
                        COLUMNS table_name VARCHAR2(200) PATH '@name') t
          WHERE index_name = 'XBRL$INSTANCEIDX') AND i.index_name NOT LIKE 'SYS%'
  ORDER BY 1, 2;
```

Example 4–12 lists the partitions and their tablespaces for partitioned XMLType table ORA$XBRLINSTANCE.

Example 4–12 Partitions and Tablespaces for XMLType Table ORA$XBRLINSTANCE

```
SELECT partition_name, tablespace_name FROM USER_TAB_PARTITIONS
  WHERE table_name = 'ORA$XBRLINSTANCE';
```

Example 4–13 lists the partitions and tablespaces for the index storage tables of partitioned `XMLIndex` index `XBRL$INSTANCEIX`.

Example 4–13 Partitions and Tablespaces for Storage Tables of XBRL$INSTANCEIX

```
SELECT y.table_name, y.partition_name, y.tablespace_name
  FROM USER_TAB_PARTITIONS y
  WHERE y.table_name
    IN (SELECT DISTINCT t.table_name
          FROM ALL_XML_INDEXES p,
               XMLTable('//xmltab'
                        PASSING p.parameters
                        COLUMNS table_name VARCHAR2(200) PATH '@name') t
          WHERE index_name ='XBRL$INSTANCEIDX')
  ORDER BY 1,2;
```

Example 4–14 lists the partitions and their tablespaces for a given partitioned index storage table—in this case, `ORAXBRL_INSTANCE_ITEM`.

Example 4–14 Partitions and Tablespaces of a Given Partitioned Index Storage Table

```
SELECT partition_name, tablespace_name FROM USER_TAB_PARTITIONS
  WHERE table_name = 'ORAXBRL_INSTANCE_ITEM';
```

Example 4–15 lists the index storage tables for partitioned `XMLIndex` index `XBRL$INSTANCEIX`, along with their secondary indexes, their partitions, and their tablespaces.

Example 4–15 Detailed Information About Partitioned Index XBRL$INSTANCEIX

```
SELECT u.table_name, i.index_name, i.partition_name, i.tablespace_name
  FROM user_ind_partitions i, user_indexes u
  WHERE i.index_name = u.index_name
    AND u.table_name
        IN (SELECT DISTINCT t.table_name
              FROM ALL_XML_INDEXES p,
                   XMLTable('//xmltab'
                            PASSING p.parameters
                            COLUMNS table_name VARCHAR2(200) PATH '@name') t
              WHERE index_name ='XBRL$INSTANCEIDX')
    AND i.index_name NOT LIKE 'SYS%'
  ORDER BY 1, 2, 3;
```

5

Installing XBRL Extension to Oracle XML DB

This chapter explains how to install XBRL Extension to Oracle XML DB.

It covers these topics:

- Hardware and Software Requirements
- Installing and Uninstalling XBRL Extension to Oracle XML DB
- Directory xbrl_xdb
- Installing a Third-Party XBRL Processing Engine
- Integration with Oracle Business Intelligence Suite

Note: Refer to "Placeholders in Oracle Database XBRL Extension Developer's Guide" on page x for explanations of the placeholders used here.

Hardware and Software Requirements

This section describes the minimal hardware and software requirements for installation and use of XBRL Extension to Oracle XML DB.

Hardware Requirements

There are no extra hardware requirements for XBRL Extension to Oracle XML DB, beyond those for Oracle Database.

Software Requirements

- Oracle Database 11g Release 2 Enterprise Edition on one of the following platforms:
  - Linux Enterprise Distribution, from Oracle or Red Hat (either 32-bit or 64-bit)
  - Microsoft Windows (either 32-bit or 64-bit)
  - Oracle Solaris Sparc or Solaris on x86-64 (64-bit)

  `http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html`
- Database patches as described in the XBRL Extension to Oracle XML DB `README.txt` file. See "Preparing to Install XBRL Extension to Oracle XML DB" on page 5-2.

- (Required for demo only.) Oracle Business Intelligence Enterprise Edition (BI-EE) 10.1.3.3, for dashboards and reports. BI-EE is not available on 64-bit platforms.

  http://www.oracle.com/technetwork/middleware/bi-enterprise-edition/downloads/business-intelligence-10g-165415.html

- Third-party XBRL processing engine (XPE) and taxonomy design tools.

Installing and Uninstalling XBRL Extension to Oracle XML DB

This section describes how to install and uninstall XBRL Extension to Oracle XML DB, create, drop, and purge an XBRL repository, and install the sample XBRL repository and GAAP demo.

Preparing to Install XBRL Extension to Oracle XML DB

This section outlines preparatory instructions for installing XBRL Extension to Oracle XML DB and related software.

1. Download the zip archive for the current release of XBRL Extension to Oracle XML DB from http://support.oracle.com. To find the file name of this zip archive, please visit the Oracle Technology Network XBRL site, http://www.oracle.com/technetwork/database/features/xmldb/index-087631.html.
2. Extract the contents of the zip archive to a temporary directory, *patch_top*.
3. Follow the instructions in *patch_top*/XBRLReleaseN/README.txt to create directory *ORACLE_HOME*/rdbms/xbrl_xdb.

   See Also: "Directory xbrl_xdb" on page 5-6 for information about the contents of directory xbrl_xdb.

Installing XBRL Extension to Oracle XML DB

Installing XBRL Extension to Oracle XML DB creates database user (schema) XBRLSYS, and it creates Oracle XML DB Repository folder /xbrl as a child of the repository root.

1. Create an Oracle Database with character set AL32UTF8.
2. Set the COMPATIBLE parameter to at least 11.2.0.1.0.
3. Set SHARED_POOL_SIZE to 1G.
4. Set the tablespace size to at least 3.5 times the size of the data on your file system, for indexed storage. For example, if the data size is 20G then set the tablespace size to at least 70G, for indexed storage.
5. Go to directory xbrl_xdb/XBRLScripts.

   ```
   shell>[1] cd XBRLScripts
   ```

6. Create a tablespace and a temporary tablespace for database user XBRLSYS.

   ```
   SQL+>[2] CREATE TABLESPACE xb_sys_ts . . .;
   SQL+> CREATE TABLESPACE xb_sys_tmp_ts  . . .;
   ```

[1] Shell examples are indicated here using the prompt shell>.

[2] The SQL examples here assume you are using SQL*Plus, for consistency. The prompt is shown as SQL+>, and the continuation prompt is shown as >. Hyphen (-) is the SQL*Plus line continuation character.

7. Run script `xbrlinstall.sql` to install XBRL Extension to Oracle XML DB.

```
SQL+> @xbrlinstall.sql sys_pass xb_sys_pass -
    >                  xb_sys_ts xb_sys_tmp_ts xb_protocols
```

8. Check the results of following SQL statements, to verify that XBRL Extension to Oracle XML DB has been successfully installed. The expected results are shown here.

```
SQL+> CONNECT XBRLSYS/xb_sys_pass
SQL+> SELECT OBJECT_NAME FROM USER_OBJECTS -
    >   WHERE STATUS = 'VALID' AND OBJECT_TYPE = 'PACKAGE';

OBJECT_NAME
---------------------
DBMS_ORAXBRL_INTERNAL
DBMS_ORAXBRLV
DBMS_ORAXBRLD
DBMS_ORAXBRL_UBM
DBMS_ORAXBRLI
DBMS_ORAXBRLT
DBMS_ORAXBRL

7 rows selected.

SQL+> SELECT ANY_PATH FROM RESOURCE_VIEW WHERE equals_path(RES, '/xbrl') = 1;

ANY_PATH
--------
/xbrl

SQL+> SELECT INDEX_NAME FROM USER_INDEXES -
    >   WHERE INDEX_TYPE = 'FUNCTION-BASED DOMAIN' AND STATUS = 'VALID';

INDEX_NAME
----------------
XBRL$SCHEMAIDX
XBRL$INSTANCEIDX
ORA$XBRLCACHEIDX

SQL+> SELECT TABLE_NAME FROM USER_OBJECT_TABLES WHERE STATUS = 'VALID';

TABLE_NAME
----------------
ORA$XBRLINSTANCE
ORA$XBRLLINKBASE
ORA$XBRLSCHEMA
```

9. If there are any error messages in log file `xbrlinstall.log`, or if the result returned by any of the SQL queries in step 8 is not as expected, then check parameter `COMPATIBLE` and run uninstall—see "UnInstalling XBRL Extension to Oracle XML DB" on page 5-4. Resolve the error, then try installing again (repeat steps 7 and 8).

Determining the Installed Version of XBRL Extension to Oracle XML DB

Use the following query to determine the current version of XBRL Extension to Oracle XML DB.

```
SELECT VALUE FROM XBRLSYS.ora$xbrlrepprop WHERE NAME='XBRLVERSION';
```

UnInstalling XBRL Extension to Oracle XML DB

Perform the following steps to uninstall XBRL Extension to Oracle XML DB. This drops all procedures and system objects created under database user `XBRLSYS`, and it deletes folder `/xbrl` from Oracle XML DB Repository.

1. Drop *each* XBRL repository – see "Dropping an XBRL Repository" on page 5-5.
2. Run SQL script `xbrluninstall.sql`.

```
SQL+> @xbrluninstall.sql sys_pass xb_sys_pass
```

3. Check the results of the following SQL statements, to verify that XBRL Extension to Oracle XML DB has successfully been uninstalled.

```
SQL+> CONNECT SYSTEM/sys_pass
SQL+> SELECT 1 FROM DBA_USERS WHERE USERNAME = 'XBRLSYS';

no rows selected

SQL+> SELECT ANY_PATH FROM RESOURCE_VIEW WHERE equals_path(RES, '/xbrl') = 1;

no rows selected
```

Creating an XBRL Repository

Perform the following steps to create an XBRL repository. This also creates a database user (schema) with the same name as the repository. You can create any number of XBRL repositories. The repositories are independent of each other.

1. Create a tablespace and a temporary tablespace for the XBRL repository. Use a redundancy factor of about 3.5 when calculating tablespace size, to account for indexed storage.

```
SQL+> CREATE TABLESPACE xb_rep_ts . . .;
SQL+> CREATE TABLESPACE xb_rep_tmp_ts . . .;
```

2. (Optional) If you want to partition the table that stores instance documents, then edit the `CREATE TABLE` statement for table `ORA$XBRLINSTANCE` in script `xbrlddl.sql` to add a virtual-column partition. See "Partitioning XBRL Repositories" on page 4-2.
3. Run SQL script, `xbrlcrt.sql`, to create the XBRL repository.

```
shell> cd XBRLScripts

SQL+> @xbrlcrt.sql sys_pass xb_sys_pass -
    >              xb_rep xb_rep_pass xb_rep_ts xb_rep_tmp_ts xb_rep_idx_ts
```

   This creates all of the tables and indexes that are needed for XBRL document storage. It also creates an Oracle XML DB Repository folder, under folder `/xbrl`, that has the same name as the XBRL repository, *xb_rep*.

4. Check the results of the following SQL statements, to verify that the repository creation was successful. The expected results are shown here.

```
SQL+> CONNECT xb_rep/xb_rep_pass
SQL+> SELECT OBJECT_NAME FROM USER_OBJECTS -
    >   WHERE STATUS = 'VALID' AND OBJECT_TYPE = 'PACKAGE';

OBJECT_NAME
--------------------
XBRL_ASYNC_EVENTS
```

```
DBMS_ORA_XBRL_EVENTS

SQL+> SELECT ANY_PATH FROM RESOURCE_VIEW
    >   WHERE equals_path(RES, '/xbrl/xb_rep') = 1;

ANY_PATH
------------
/xbrl/xb_rep

SQL+> SELECT INDEX_NAME FROM USER_INDEXES -
    >   WHERE INDEX_TYPE = 'FUNCTION-BASED DOMAIN' AND STATUS = 'VALID';

INDEX_NAME
----------------
ORA$XBRLCACHEIDX
XBRL$INSTANCEIDX
XBRL$SCHEMAIDX

SQL+> SELECT TABLE_NAME FROM USER_OBJECT_TABLES WHERE STATUS = 'VALID';

TABLE_NAME
----------------
ORA$XBRLINSTANCE
ORA$XBRLLINKBASE
ORA$XBRLSCHEMA
```

Check for any error messages in log file `xbrlcft.log`.

5. If there are any error messages in log file `xbrlcft.log`, or if the result returned by any of the SQL queries in step 4 is not as expected, then run script `xbrldrop.sql` to drop the newly created repository—see "Dropping an XBRL Repository" on page 5-5. Resolve the error, then create the repository again (repeat steps 3 and 4).

Dropping an XBRL Repository

Perform the following steps to drop (delete) an XBRL repository.

1. Run SQL script `xbrldrop.sql`.

```
SQL+> @xbrldrop.sql xb_sys_pass xb_rep xb_rep_pass
```

2. Check the results of the following SQL statements, to verify that the drop was successful.

```
SQL+> CONNECT SYSTEM/sys_pass
SQL+> SELECT 1 FROM DBA_USERS WHERE USERNAME = UPPER ('xb_rep');

no rows selected

SQL+> SELECT ANY_PATH FROM RESOURCE_VIEW
    >   WHERE equals_path(RES, '/xbrl/xb_rep') = 1;

no rows selected
```

Purging an Accidentally Dropped XBRL Repository

In case the database schema corresponding to an XBRL repository is dropped, you can perform the following steps to purge an XBRL repository, deleting all dependent objects that were created in Oracle XML DB Repository.

1. Run SQL script `xbrludpurge.sql`.

```
SQL+> @xbrludpurge.sql xb_sys_pass xb_rep
```

2. Check the results of the following SQL statements, to verify that the objects corresponding to the XBRL repository have been successfully purged.

```
SQL+> CONNECT SYSTEM/sys_pass
SQL+> SELECT ANY_PATH FROM RESOURCE_VIEW
    >   WHERE equals_path(RES, '/xbrl/xb_rep') = 1;

no rows selected
```

Installing the Sample XBRL Repository and GAAP Demo

Perform the following steps to install the sample XBRL repository and GAAP demo.

1. Run SQL script `InstallXBRLDemo.sql` from directory `xbrl_xdb/XBRLScripts`. This creates a sample XBRL repository named `oraxbrl`.
2. Follow the instructions in "Building and Using a Sample XBRL Application: USGAAP 2008" on page 2-4.
3. Refer to the SQL statements in `xbrl_xdb/XBRLScripts`, to become familiar with the APIs of XBRL Extension to Oracle XML DB.

Directory xbrl_xdb

This section describes the contents of the `xbrl_xdb` directory.

XBRLScripts

Directory `XBRLScripts` contains the following SQL script files.

Table 5–1 SQL Scripts in Directory XBRLScripts

SQL Script Name	Description
`xbrlinstall.sql`	Create database user `XBRLSYS` and install packages that contain XBRL- specific APIs. If Oracle XML DB Repository is used, then a root directory `/xbrl` is created, with owner `XBRLSYS`.
`xbrlcrt.sql`	Create an XBRL repository and a database schema with the same name. Create all necessary tables, indexes, and procedures.
`xbrlddl.sql`	Create tables and indexes. This script is run automatically by script `xbrlcrt.sql`. If you want to partition the base table for the XBRL instance documents of a given repository, then edit script `xbrlddl.sql` before you use script `xbrlcrt.sql` to create that repository. See "Partitioning XBRL Repositories" on page 4-2.
`xbrldrop.sql`	Drop a given XBRL repository, including the corresponding database schema, tables, indexes, and procedures.
`xbrluninstall.sql`	Finish uninstalling. Invoke this after dropping all XBRL repositories.
`xbrludpurge.sql`	Remove other system objects associated with an XBRL repository. Use this if a database schema corresponding to an XBRL repository is dropped accidentally.
`xbrlpurgefile.sql`	Delete resources from Oracle XML DB Repository that are associated with an XBRL repository. Use this if you use Oracle XML DB Repository and you mistakenly delete a document from the XBRL repository.

Table 5–1 (Cont.) SQL Scripts in Directory XBRLScripts

SQL Script Name	Description
`xbrlrecidxdrv.sql`	Drop the `XMLIndex` indexes used for an XBRL repository, then recreate them, so you can move the index storage tables to a different tablespace.
`xbrlregschema.sql`	Register standard XBRL schemas. Must be run before using the tuple APIs.
`xbrlerrmsg.sql`	Load error messages in different languages for XBRL Extension to Oracle XML DB.
`InstallXBRLDemo.sql`	Install XBRL Extension to Oracle XML DB and create an XBRL repository.

XBRLDemoScripts

Directory `XBRLDemoScripts` contains the following files.

- `demo.sql` – Script that loads and queries Bank of America and USGAAP 2008 files.
- `demo2.sql` – Script that loads and queries a tuple demo (files `oraclexbrltupledemo.xsd`, `oraclexbrltupledemo-inst.xml`).
- Files `schema.xml` and `linkbase.xml`, which are used by the USGAAP 2008 demo. These files list the XBRL schema and linkbase files in USGAAP 2008. See "Building and Using a Sample XBRL Application: USGAAP 2008" on page 2-4.

Demo-BIFiles

Directory `Demo-BIFiles` contains Oracle Business Intelligence Suite Enterprise Edition (OBIEE) resources used for the USGAAP 2008 demo. It contains these subdirectories:

- `xbrl.rpd` – Sample business intelligence XBRL application based on USGAAP 2008.
- `xbrl.zip` – Sample business intelligence XBRL application based on USGAAP 2008.

Installing a Third-Party XBRL Processing Engine

You must install an Oracle-certified third-party XBRL processing engine, outside the database. For information about this, consult Oracle XBRL support:

http://www.oracle.com/technetwork/database/features/xmldb/index-087631.html

Integration with Oracle Business Intelligence Suite

Oracle Business Intelligence Suite Enterprise Edition (OBIEE) is not included as part of XBRL Extension to Oracle XML DB. You must procure it separately and install it according to the OBIEE instructions. You can install it in any tier. You must install an Oracle client for OBIEE to work properly with XBRL Extension to Oracle XML DB.

XBRL Extension to Oracle XML DB provides a demo package to demonstrate integration with OBIEE. Included in directory `xbrl_xdb` is a directory `Demo-BIFiles`, which contains the OBIEE repository file `xbrl.rpd` and folders and files for a sample dashboard, in zip archive `xbrl.zip`.

See Also:

- http://st-curriculum.oracle.com/obe/fmw/bi/biee/r1013/bi_admin/biadmin.html – Building and managing an Oracle Business Intelligence repository
- http://st-curriculum.oracle.com/obe/fmw/bi/biee/r1013/saw/saw.html – Oracle Business Intelligence queries and interactive dashboards

Perform the following steps to configure OBIEE with the demo package:

1. Establish a connection to the database instance.
   1. Add entry `DEMO` to file *`oracle_client_dir`*`\network\admin\tnsnames.ora`. In the following, change *`hostname.domain`*, *`port`*, and *`sid`* to the correct values for your database instance:

      ```
      DEMO = (DESCRIPTION =
               (ADDRESS_LIST =
                 (ADDRESS = (PROTOCOL = TCP)
                            (HOST = hostname.domain)
                            (PORT = port)))
               (CONNECT_DATA = (SERVICE_NAME = sid)))
      ```

   2. Select from the Microsoft Windows **Start** menu: *`oracle_client`*, then **Configuration and Migration Tools**, then **Microsoft ODBC Administrator**.
   3. In the **Oracle ODBC Driver Configuration** dialog box, click the **System DSN** tab. Click **Add**. Select *`oracle_client`* as the driver. Click **Finish**.
   4. Enter `x02` as the **Data Source Name**, `demo` as the **TNS Service Name**, and `oraxbrl` as the **User ID**. Click **OK**.
2. Copy OBIEE repository file `xbrl.rpd` from `xbrl_xdb\Demo-BIFiles` to *`obiee_home`*`\server\Repository`.
3. Extract zip archive `xbrl.zip` to directory *`obieedata_home`*`\web\catalog`.
4. Open file *`obiee_home`*`\server\Config\NQSConfig.INI` in a text editor, and change the text that follows `Star =` to make it `xbrl.rpd`:

   ```
   [ REPOSITORY ]
   Star = xbrl.rpd, DEFAULT;
   ```

5. Open file *`obieedata_home`*`\web\config\instanceconfig.xml` in a text editor, and change the final directory component of the catalog path to make it `xbrl`:

   ```
   <CatalogPath>obieedata_home/web/catalog/xbrl</CatalogPath>
   ```

6. Start the Oracle BI services.
   1. Choose from the Microsoft Windows **Start** menu: **Run**.
   2. Enter `services.msc`.
   3. In the **Services** dialog box, start or restart each of the following services, by selecting the service name, right-clicking, and then selecting **Start** or **Restart**: **Oracle BI Server**, **Oracle BI Javahost**, and **Oracle BI Presentation Server**.

Index

Numerics

A

B

C

D

G

I

L

M

O

P

R

S

T

U

V

X