240
DBMS_XMLSTORAGE_MANAGE

The DBMS_XMLSTORAGE_MANAGE package provides an interface to manage and modify XML storage after schema registration has been completed.

This chapter contains the following topics:

- Overview
- Security Model
- Summary of DBMS_XMLSTORAGE_MANAGE Subprograms

> **See Also:**
>
> *Oracle XML DB Developer's Guide*

DBMS_XMLSTORAGE_MANAGE Overview

DBMS_XMLSTORAGE_MANAGE contains procedures to manage and modify XML storage after schema registration has been completed.

Use subprograms from this package to improve the performance of bulk load operations. You can disable indexes and constraints before doing a bulk load process and to enable them afterwards.

DBMS_XMLSTORAGE_MANAGE Security Model

Owned by XDB, the DBMS_XMLSTORAGE_MANAGE package must be created by SYS or XDB. The EXECUTE privilege is granted to PUBLIC.

Subprograms in this package are executed using the privileges of the current user.

Summary of DBMS_XMLSTORAGE_MANAGE Subprograms

This table lists and describes the DBMS_XML_STORAGE package subprograms.

Table 240-1 DBMS_XMLSTORAGE_MANAGE Package Subprograms

Subprogram	Description
DISABLEINDEXESANDCONSTRAINTS Procedure	Disables the indexes and constraints for XMLType tables and XMLType columns
ENABLEINDEXESANDCONSTRAINTS Procedure	Rebuilds all indexes and enables the constraints on an XMLType table including its child tables and out-of-line tables
EXCHANGEPOSTPROC Procedure	Enable constraints after exchange partition

Table 240-1 (Cont.) DBMS_XMLSTORAGE_MANAGE Package Subprograms

Subprogram	Description
EXCHANGEPREPROC Procedure	Disable constraints before exchange partition
INDEXXMLREFERENCES Procedure	Creates unique indexes on the `REF` columns of the given XML type table or the XML type column of a given table
REFPARTITIONEXCHANGEIN Procedure	This is an auxiliary procedure to load data through exchange partition operation into a partitioned table and its reference-partitioned child table provided the child table has an `xmltype` column with a local `xmlindex`.
REFPARTITIONEXCHANGEOUT Procedure	Auxiliary procedure to load data through exchange partition operation out of a partitioned table and its reference-partitioned child table provided that the child table has an `xmltype` column with a local `xmlindex`.
RENAMECOLLECTIONTABLE Procedure	Renames a collection table to the given table name
SCOPEXMLREFERENCES Procedure	Scopes all XML references. Scoped `REF` types require less storage space and allow more efficient access than unscoped `REF` types
XPATH2TABCOLMAPPING Function	Maps a path expression (in XPath notation or DOT notations) to the corresponding table name and column name

DISABLEINDEXESANDCONSTRAINTS Procedure

This procedure disables the indexes and constraints for `XMLType` tables and `XMLType` columns.

Syntax

```
DBMS_XMLSTORAGE_MANAGE.DISABLEINDEXESANDCONSTRAINTS (
   owner_name    IN  VARCHAR2 DEFAULT USER,
   table_name    IN  VARCHAR2,
   column_name   IN  VARCHAR2 DEFAULT NULL,
   clear         IN  BOOLEAN  DEFAULT FALSE);
```

Parameters

Table 240-2 DISABLEINDEXESANDCONSTRAINTS Procedure Parameters

Parameter	Description
`owner_name`	Owner's name
`table_name`	Name of the `XMLType` table that the procedure is being performed on
`column_name`	`XMLType` column name
`clear`	Boolean that when set to `TRUE` clears all stored index and constraint data for the table before the procedure executes. The default is `FALSE`, which does not clear them.

Usage Notes

Passing XMLTYPE tables

For XMLType tables, you must pass the XMLType table name on which the bulk load operation is to be performed. For XMLType columns, you must pass the relational table name and the corresponding XMLType column name.

Using clear to Enable and Disable Indexes and Constraints

> **Note:**
>
> If the DISABLEINDEXESANDCONTRAINTS procedure is called with clear set to TRUE, it removes any index or constraint information about the XMLTYPE table or column memorized during earlier executions of the procedure.

Therefore, you must ensure that all disabled indexes and constraints are re-enabled on the table or column before you call the DISABLEINDEXESANDCONTRAINTS procedure with clear set to TRUE.

Ideally, it is recommended that you set clear set to TRUE for the first execution. For any subsequent executions (due to errors while disabling or enabling indexes) clear should be set to FALSE, the default value. Once you have successfully re-enabled all the indexes and constraints following the bulk load operation, you can call this procedure again with clear set to TRUE for the next bulk load operation.

Example

The following example illustrates the use of clear in the DISABLEINDEXESANDCONSTRAINTS procedure and the ENABLEINDEXESANDCONSTRAINTS Procedure.

First, add a not-NULL constraint on comment element of the PURCHASEORDER_TAB table:

```
ALTER TABLE PURCHASEORDER_TAB ADD CONSTRAINT c1 check            ("XMLDATA"."comment" IS
NOT NULL);
```

Then, disable all the indexes and constraints by passing the clear as TRUE, by calling the DISABLEINDEXESANDCONSTRAINTS procedure:

```
BEGIN
     XDB.DBMS_XMLSTORAGE_MANAGE.DISABLEINDEXESANDCONSTRAINTS
                                ( USER,'PURCHASEORDER_TAB',NULL,TRUE );
END;
/
```

Next, perform a bulk load operation (such as datapump import) which violates constraint c1 in the ALTER table statement. This does not raise an error because the constraint is disabled:

```
host impdp orexample/orexample directory=dir dumpfile=dmp.txt
   tables=OREXAMPLE.PURCHASEORDER_TAB content = DATA_ONLY;
```

NOTE: To view the disabled constraints and indexes use:

```
SELECT constraint_name,table_name,status FROM all_constraints
  WHERE owner = user;
```

Finally, try to enable the constraint using the ENABLEINDEXESANDCONSTRAINTS procedure. It raises an error because c1, the not null constraint, is violated by the bulk load operation:

```
BEGIN
     XDB.DBMS_XMLSTORAGE_MANAGE.ENABLEINDEXESANDCONSTRAINTS
                                          ( USER,'PURCHASEORDER_TAB');
END;
/
```

To disable all the indexes and constraints, again use `DISABLEINDEXESANDCONSTRAINTS`, but set `clear= FALSE` (because the `ENABLEINDEXESANDCONSTRAINTS` failed to complete successfully). Note: `clear = FALSE` by default, so we do not need to pass it explicitly in the next call.

```
BEGIN
     xdb.DBMS_XMLSTORAGE_MANAGE.DISABLEINDEXESANDCONSTRAINTS
                                ( USER,'PURCHASEORDER_TAB');
END;
/
```

Then, delete the incorrect rows entered into the table

```
DELETE FROM purchaseorder_tab p
     WHERE p.xmldata."comment" IS NULL;
```

Re-enable the indexes and constraints using `ENABLEINDEXESANDCONSTRAINTS`, which completes successfully.

```
BEGIN
     xdb.DBMS_XMLSTORAGE_MANAGE.ENABLEINDEXESANDCONSTRAINTS
                                          ( USER,'PURCHASEORDER_TAB');
END;
/
```

ENABLEINDEXESANDCONSTRAINTS Procedure

This procedure rebuilds all indexes and enables the constraints on an `XMLType` table including its child tables and out-of-line tables.

When `column_name` is passed, it does the same for this `XMLType` column.

Syntax

```
DBMS_XMLSTORAGE_MANAGE.ENABLEINDEXESANDCONSTRAINTS (
   owner_name   IN VARCHAR2 DEFAULT USER,
   table_name   IN VARCHAR2,
   column_name  IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 240-3 ENABLEINDEXESANDCONSTRAINTS Procedure Parameters

Parameter	Description
`owner_user`	Owner's name
`table_name`	Name of the table that the indexes and constraints are being removed from
`column_name`	Column name

Usage Notes

This procedure reverses DISABLEINDEXESANDCONSTRAINTS Procedure.

Example

See DISABLEINDEXESANDCONSTRAINTS Procedure

EXCHANGEPOSTPROC Procedure

This procedure enable constraints after exchange partition.

Syntax

```
DBMS_XMLSTORAGE_MANAGE.EXCHANGEPOSTPROC (
   owner_name   IN VARCHAR2 DEFAULT USER,
   table_name   IN VARCHAR2);
```

Parameters

Table 240-4 EXCHANGEPOSTPROC Procedure Parameters

Parameter	Description
owner_user	Owner's name
table_name	Name of the table that the indexes and constraints are being removed from

EXCHANGEPREPROC Procedure

This procedure disable constraints before exchange partition.

Syntax

```
DBMS_XMLSTORAGE_MANAGE.EXCHANGEPREPROC (
   owner_name   IN VARCHAR2 DEFAULT USER,
   table_name   IN VARCHAR2);
```

Parameters

Table 240-5 EXCHANGEPREPROC Procedure Parameters

Parameter	Description
owner_user	Owner's name
table_name	Name of the table that the indexes and constraints are being removed from

INDEXXMLREFERENCES Procedure

This procedure creates unique indexes on the `REF` columns of the given XML type table or the XML type column of a given table.

If the procedure creates multiple `REF` columns, it appends `_1`, `_2`, and so on to their names.

Syntax

```
DBMS_XMLSTORAGE_MANAGE.INDEXXMLREFERENCES (
   owner_name     IN VARCHAR2 DEFAULT USER,
   table_name     IN VARCHAR2,
   column_name    IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 240-6 INDEXXMLREFERENCES Procedure Parameters

Parameter	Description
owner_name	The owner's name
table_name	The table being indexed
column_name	A column name. Not needed for XML type tables.
index_name	The name of the newly created index

Usage Notes

This procedure is only used if the REFs are scoped. See SCOPEXMLREFERENCES Procedure.

Indexed REFs lead to better performance when joins between the base table and a child table occur in the query plan.

- If the base table has a higher selectivity than the child table, there is no need to index the REFs.
- If the selectivity of the child table is higher than that of the base table and if no indexes are present, then the join of one row in the child table with the base table leads to a full table scan of the base table.

INDEXXMLREFERENCES does not index REFs recursively in child tables of a table it is called on. To do this, Oracle recommends calling the procedure from within a loop over the XML_OUT_OF_LINE_TABLES or XML_NESTED_TABLES view. This creates the index names from the current value of a column in the view.

> **Note:**
>
> This procedure is limited to the structured storage model.

REFPARTITIONEXCHANGEIN Procedure

This is an auxiliary procedure to load data through exchange partition operation into a partitioned table and its reference-partitioned child table provided that the child table has an xmltype column with a local xmlindex.

Syntax

```
DBMS_XMLSTORAGE_MANAGE.REFPARTITIONEXCHANGEIN (
   owner_name                IN VARCHAR2,
   parent_table_name         IN VARCHAR2,
```

```
   child_table_name           IN VARCHAR2,
   parent_exchange_table_name IN VARCHAR2,
   child_exchange_table_name  IN VARCHAR2,
   parent_exchange_stmt       IN CLOB,
   child_exchange_stmt        IN CLOB);
```

Parameters

Table 240-7 REFPARTITIONEXCHANGEIN Parameters

Parameter	Description
owner_name	owner's name
parent_table_name	the partitioned base table
child_table_name	a partitioned table with reference partitioning based on the table named parent_table_name
parent_exchange_table_name	an exchange table for the partitioned base table
child_exchange_table_name	an exchange table for the table named child_table_name
parent_exchange_stmt	SQL statement to execute exchange partition operation between the table named parent_table_name and the table named parent_exchange_table_name
child_exchange_stmt	SQL statement to execute exchange partition operation between the table named child_table_name and the table named child_exchange_table_name

REFPARTITIONEXCHANGEOUT Procedure

This is an auxiliary procedure to load data through exchange partition operation out of a partitioned table and its reference-partitioned child table provided that the child table has an xmltype column with a local xmlindex.

Syntax

```
DBMS_XMLSTORAGE_MANAGE.REFPARTITIONEXCHANGEOUT (
   owner_name                 IN VARCHAR2,
   parent_table_name          IN VARCHAR2,
   child_table_name           IN VARCHAR2,
   parent_exchange_table_name IN VARCHAR2,
   child_exchange_table_name  IN VARCHAR2,
   parent_exchange_stmt       IN CLOB,
   child_exchange_stmt        IN CLOB);
```

Parameters

Table 240-8 REFPARTITIONEXCHANGEOUT Parameters

Parameter	Description
owner_name	owner's name

Table 240-8 (Cont.) REFPARTITIONEXCHANGEOUT Parameters

Parameter	Description
parent_table_name	the partitioned base table
child_table_name	a partitioned table with reference partitioning based on the table named parent_table_name
parent_exchange_table_name	an exchange table for the partitioned base table
child_exchange_table_name	an exchange table for the table named child_table_name
parent_exchange_stmt	SQL statement to execute exchange partition operation between the table named parent_table_name and the table named parent_exchange_table_name
child_exchange_stmt	SQL statement to execute exchange partition operation between the table named child_table_name and the table named child_exchange_table_name

RENAMECOLLECTIONTABLE Procedure

This procedure renames a collection table to the given table name.

An XPath expression specifies the collection table, starting from the XMLtype base table or an XMLType column of the base table.

This procedure provides the only way to derive a collection table name from the corresponding collection type name because there is no direct schema annotation for the purpose.

Syntax

```
DBMS_XMLSTORAGE_MANAGE.RENAMECOLLECTIONTABLE (
   owner_name             IN VARCHAR2 DEFAULT USER,
   table_name             IN VARCHAR2,
   column_name            IN VARCHAR2 DEFAULT NULL,
   xpath                  IN VARCHAR2,
   collection_table_name  IN VARCHAR2
   namespaces             IN VARCHAR2 default NULL);  // For release 11.2 only
```

Parameters

Table 240-9 RENAMECOLLECTIONTABLE Procedure Parameters

Parameter	Description
owner_name	The name of the owner
table_name	The name of a base table that can be used as the starting point for specifying the collection table
column_name	An XMLType column that can be the starting point for specifying the collection table
xpath	The XPath expression that specifies the collection table
collection_table_name	The name of the collection table

Table 240-9 (Cont.) RENAMECOLLECTIONTABLE Procedure Parameters

Parameter	Description
namespaces	For Oracle Database 11*g* Release 2 (11.2) and higher. The namespaces used in XPath.

Usage Notes

Call this procedure after registering the XML schema.

The table name serves as a prefix to the index names.

Oracle recommends using this function because it makes query execution plans more readable.

Report errors that occur while this procedure runs to the user that called the procedure.

> **Note:**
>
> This procedure is limited to the structured storage model.

For Oracle Database 11*g* Release 2 (11.2) and higher, only, this function accepts XPath notation as well as DOT notation. If XPath notation is used, a `namespaces` parameter may also be required.

Example

The collection table name will be `EMP_TAB_NAMELIST`. You can verify this using `SELECT * FROM user_nested_tables`.

Using DOT Notation:

```
call  XDB.DBMS_XMLSTORAGE_MANAGE.RENAMECOLLECTIONTABLE (
            USER,
            'EMP_TAB',
            NULL,
            '"XMLDATA"."EMPLOYEE"."NAME"',
           ''EMP_TAB_NAMELIST);
```

Using XPath Notation:

XPath notation is available with Oracle Database 11g Release 2 (11.2) and higher.

```
call  XDB.DBMS_XMLSTORAGE_MANAGE.RENAMECOLLECTIONTABLE (
               USER,
               'EMP_TAB',
               NULL,
               '/e:Employee/Name',
               'EMP_TAB_NAMELIST',
               '''http://www.oracle.com/emp.xsd'' as "e"');
```

SCOPEXMLREFERENCES Procedure

This procedure scopes all XML references. Scoped `REF` types require less storage space and allow more efficient access than unscoped `REF` types.

Syntax

```
DBMS_XMLSTORAGE_MANAGE.SCOPEXMLREFERENCES;
```

Usage Notes

- If you have used SETOUTOFLINE Procedure in the DBMS_XMLSTORAGE_MANAGE package to avoid raising `'4096 column limit'` errors during XML schema registration, you should also use SCOPEXMLREFERENCES Procedure.
- Using `SCOPEXMLREFERENCES` after XML schema registration and before loading XML instance data, makes these reference scoped to the out-of-line table only.

> **Note:**
>
> This procedure is limited to the structured storage model.

XPATH2TABCOLMAPPING Function

This function maps a path expression (in XPath notation or DOT notations) to the corresponding table name and column name. This is necessary in cases in which the user wants to create an index on this table, or to add a constraint, or to rename a table to make query execution plans more readable.

Syntax

```
DBMS_XMLSTORAGE_MANAGE.XPATH2TABCOLMAPPING (
   owner_name    IN  VARCHAR2 DEFAULT USER,
   table_name    IN  VARCHAR2,
   column_name   IN  VARCHAR2 DEFAULT NULL,
   xpath         IN  VARCHAR2,
   namespaces    IN  VARCHAR2 DEFAULT NULL)
 RETURN XMLTYPE;
```

Parameters

Table 240-10 XPATH2TABCOLMAPPING Procedure Parameters

Parameter	Description
`owner_user`	Owner's name
`table_name`	Name of the base table
`column_name`	Optional name of the XML type column if `table_name` is not an XMLtype table. If table_name refers to XMLType table then `column_name` should be NULL.
`xpath`	Path expression in DOT notation or XPath notation (see examples below)

Table 240-10 (Cont.) XPATH2TABCOLMAPPING Procedure Parameters

Parameter	Description
namespaces	Optional namespace definitions for path expression

Examples

XPath2TablColMapping evaluated on XMLType table with Xpath Notation, namespaces provided

```
SELECT XDB.DBMS_XMLSTORAGE_MANAGE.XPATH2TABCOLMAPPING (
   USER, 'XML_TAB', '', '//n1:item/n1:location',''''xdbXmark'' as "n1"'')
FROM DUAL;
```

This produces a result, for example:

```
<Result>
<Mapping TableName="SYS_NT12345" ColumnName="location"/>
</Result>
```

This allows us to define an index or constraint on table `SYS_NT12345` and column `location`.

XPath2TablColMapping evaluated on table not of XMLType but with XMLType column by means of DOT notation

```
SELECT XDB.DBMS_XMLSTORAGE_MANAGE.XPATH2TABCOLMAPPING (
   USER,'PurchaseOrderTab','XMLCOL','xmldata.LineItems.LineItem', '')
FROM DUAL;
```