12

PL/SQL Package DBMS_XMLSTORE

You can use the PL/SQL package `DBMS_XMLSTORE` to insert, update, or delete data from XML documents stored object-relationally. It uses a canonical XML mapping similar to the one produced by package `DBMS_XMLGEN`. It converts the mapping to object-relational constructs and then inserts, updates or deletes the corresponding values in relational tables.

> **Note:**
>
> The PL/SQL package `DBMS_XMLSTORE` is deprecated in Oracle Database 23ai.
>
> This package is deprecated, and can be desupported in a future release. Oracle recommends that you use regular SQL DML and with standard XQuery and SQL/XML to store and manage XML data. Using standard functionality provides future-proof way to store and manipulate XML data.

- Using Package DBMS_XMLSTORE
  Create a context handle, perform one or more insertions, updates, or deletions, and then close the context,
- Inserting an XML Document Using DBMS_XMLSTORE
  You can use PL/SQL package `DBMS_XMLSTORE` to insert an XML document into a table or view. It parses the document and creates an `INSERT` statement into which it binds the values. By default, it inserts values into all of the columns represented by elements in the XML document.
- Updating XML Data Using DBMS_XMLSTORE
  You can use PL/SQL package `DBMS_XMLSTORE` to update (modify) existing data. You specify which rows to update by calling procedure `setKeyColumn` once for *each* of the columns that are used collectively to identify the row. (In SQL, you would specify the rows using a `WHERE` clause in an `UPDATE` statement.)
- Deleting XML Data Using DBMS_XMLSTORE
  You can use PL/SQL package `DBMS_XMLSTORE` to delete existing data. You specify which rows to delete by calling procedure `setKeyColumn` once for *each* of the columns that are used collectively to identify the row. (In SQL, you would specify the rows using a `WHERE` clause in an `UPDATE` statement.)

Using Package DBMS_XMLSTORE

Create a context handle, perform one or more insertions, updates, or deletions, and then close the context,

To use PL/SQL package `DBMS_XMLSTORE`, follow these steps:

1. Create a context handle by calling function `DBMS_XMLSTORE.`**`newContext`** and supplying it with the table name to use for the DML operations. For case sensitivity, double quotation mark (") the string that is passed to the function.

By default, XML documents are expected to use the `<ROW>` tag to identify rows. This is the same default used by package `DBMS_XMLGEN` when generating XML data. You can use function `setRowTag` to override this behavior.

2. (Optional) Perform one or more insertions, updates or deletions (you can *repeat* this step):
   - For *insertions*, to improve performance you can specify the list of columns to insert by calling procedure `DBMS_XMLSTORE.setUpdateColumn` for each column. The default behavior (if you do not specify the list of columns) is to insert values for each column whose corresponding element is present in the XML document.
   - For *updates*, use function `DBMS_XMLSTORE.setKeyColumn` to specify one or more (pseudo-) key columns, which are used to specify the *rows* to update. You do this in the `WHERE` clause of a SQL `UPDATE` statement. The columns that you specify need not be keys of the table, but together they must uniquely specify the rows to update.

     For example, in table `employees`, column `employee_id` uniquely identifies rows (it is a key of the table). If the XML document that you use to update the table contains element `<EMPLOYEE_ID>2176</EMPLOYEE_ID>`, then the rows where `employee_id` equals `2176` are updated.

     To improve performance, you can also specify the list of update columns using `DBMS_XMLSTORE.setUpdateColumn`. The default behavior is to update *all* of the columns in the row(s) identified by `setKeyColumn` whose corresponding elements are present in the XML document.
   - For *deletions* you specify (pseudo-) key columns to identify the row(s) to delete. You do this the same way you specify rows to update (see previous).
3. Provide a document to `DBMS_XMLSTORE` function `insertXML`, `updateXML`, or `deleteXML`. You can *repeat* this step to update several XML documents.
4. Close the context by calling function `DBMS_XMLSTORE.closeContext`.

Inserting an XML Document Using DBMS_XMLSTORE

You can use PL/SQL package `DBMS_XMLSTORE` to insert an XML document into a table or view. It parses the document and creates an `INSERT` statement into which it binds the values. By default, it inserts values into all of the columns represented by elements in the XML document.

Example 12-1 uses `DBM_XMLSTORE` to insert the information for two new employees into the `employees` table. The information to insert is provided as XML data.

Example 12-1 Inserting Data with Specified Columns

```
SELECT employee_id AS EMP_ID, salary, hire_date, job_id, email, last_name
  FROM employees WHERE department_id = 30;

EMP_ID     SALARY HIRE_DATE JOB_ID     EMAIL       LAST_NAME
------ ---------- --------- ---------- ---------- ----------
   114      11000 07-DEC-94 PU_MAN     DRAPHEAL     Raphaely
   115       3100 18-MAY-95 PU_CLERK   AKHOO            Khoo
   116       2900 24-DEC-97 PU_CLERK   SBAIDA          Baida
   117       2800 24-JUL-97 PU_CLERK   STOBIAS        Tobias
   118       2600 15-NOV-98 PU_CLERK   GHIMURO        Himuro
   119       2500 10-AUG-99 PU_CLERK   KCOLMENA   Colmenares

6 rows selected.
```

```
DECLARE
  insCtx DBMS_XMLSTORE.ctxType;
  rows NUMBER;
  xmlDoc CLOB :=
    '<ROWSET>
       <ROW num="1">
         <EMPLOYEE_ID>920</EMPLOYEE_ID>
         <SALARY>1800</SALARY>
         <DEPARTMENT_ID>30</DEPARTMENT_ID>
         <HIRE_DATE>17-DEC-2002</HIRE_DATE>
         <LAST_NAME>Strauss</LAST_NAME>
         <EMAIL>JSTRAUSS</EMAIL>
         <JOB_ID>ST_CLERK</JOB_ID>
       </ROW>
       <ROW>
         <EMPLOYEE_ID>921</EMPLOYEE_ID>
         <SALARY>2000</SALARY>
         <DEPARTMENT_ID>30</DEPARTMENT_ID>
         <HIRE_DATE>31-DEC-2004</HIRE_DATE>
         <LAST_NAME>Jones</LAST_NAME>
         <EMAIL>EJONES</EMAIL>
         <JOB_ID>ST_CLERK</JOB_ID>
       </ROW>
     </ROWSET>';
BEGIN
  insCtx := DBMS_XMLSTORE.newContext('HR.EMPLOYEES'); -- Get saved context
  DBMS_XMLSTORE.clearUpdateColumnList(insCtx); -- Clear the update settings

  -- Set the columns to be updated as a list of values
  DBMS_XMLSTORE.setUpdateColumn(insCtx, 'EMPLOYEE_ID');
  DBMS_XMLSTORE.setUpdateColumn(insCtx, 'SALARY');
  DBMS_XMLSTORE.setUpdateColumn(insCtx, 'HIRE_DATE');
  DBMS_XMLSTORE.setUpdateColumn(insCtx, 'DEPARTMENT_ID');
  DBMS_XMLSTORE.setUpdateColumn(insCtx, 'JOB_ID');
  DBMS_XMLSTORE.setUpdateColumn(insCtx, 'EMAIL');
  DBMS_XMLSTORE.setUpdateColumn(insCtx, 'LAST_NAME');

  -- Insert the doc.
  rows := DBMS_XMLSTORE.insertXML(insCtx, xmlDoc);
  DBMS_OUTPUT.put_line(rows || ' rows inserted.');

  -- Close the context
  DBMS_XMLSTORE.closeContext(insCtx);
END;
/

2 rows inserted.

PL/SQL procedure successfully completed.

SELECT employee_id AS EMP_ID, salary, hire_date, job_id, email, last_name
  FROM employees WHERE department_id = 30;

EMP_ID     SALARY HIRE_DATE JOB_ID     EMAIL       LAST_NAME
------ ---------- --------- ---------- ---------- ----------
   114      11000 07-DEC-94 PU_MAN     DRAPHEAL     Raphaely
```

```
   115       3100 18-MAY-95 PU_CLERK   AKHOO            Khoo
   116       2900 24-DEC-97 PU_CLERK   SBAIDA           Baida
   117       2800 24-JUL-97 PU_CLERK   STOBIAS          Tobias
   118       2600 15-NOV-98 PU_CLERK   GHIMURO          Himuro
   119       2500 10-AUG-99 PU_CLERK   KCOLMENA     Colmenares
   920       1800 17-DEC-02 ST_CLERK   STRAUSS         Strauss
   921       2000 31-DEC-04 ST_CLERK   EJONES           Jones

8 rows selected.
```

Updating XML Data Using DBMS_XMLSTORE

You can use PL/SQL package `DBMS_XMLSTORE` to update (modify) existing data. You specify which rows to update by calling procedure `setKeyColumn` once for *each* of the columns that are used collectively to identify the row. (In SQL, you would specify the rows using a `WHERE` clause in an `UPDATE` statement.)

You can think of this set of columns as acting like a set of key columns: *together, they specify a unique row* to be updated. However, the columns that you use (with `setKeyColumn`) need *not* be keys of the table — as long as they uniquely specify a row, they can be used with calls to `setKeyColumn`.

Example 12-2 uses `DBM_XMLSTORE` to update information. Assuming that the first name for employee number 188 is incorrectly recorded as Kelly, this example corrects that first name to Pat. Since column `employee_id` is a primary key for table `employees`, a single call to `setKeyColumn` specifying column `employee_id` is sufficient to identify a unique row for updating.

The following `UPDATE` statement is equivalent to the use of `DBM_XMLSTORE` in Example 12-2:

```
UPDATE hr.employees SET first_name = 'Pat' WHERE employee_id = 188;
```

Example 12-2 Updating Data with Key Columns

```
SELECT employee_id, first_name FROM employees WHERE employee_id = 188;

EMPLOYEE_ID FIRST_NAME
----------- ----------
        188 Kelly

1 row selected.

DECLARE
  updCtx DBMS_XMLSTORE.ctxType;
  rows NUMBER;
  xmlDoc CLOB :=
    '<ROWSET>
       <ROW>
         <EMPLOYEE_ID>188</EMPLOYEE_ID>
         <FIRST_NAME>Pat</FIRST_NAME>
       </ROW>
     </ROWSET>';
BEGIN
   updCtx := DBMS_XMLSTORE.newContext('HR.EMPLOYEES'); -- get the context
   DBMS_XMLSTORE.clearUpdateColumnList(updCtx);        -- clear update
```

```
settings

   -- Specify that column employee_id is a "key" to identify the row to
update.
   DBMS_XMLSTORE.setKeyColumn(updCtx, 'EMPLOYEE_ID');
   rows := DBMS_XMLSTORE.updateXML(updCtx, xmlDoc);    -- update the table
   DBMS_XMLSTORE.closeContext(updCtx);                 -- close the context
END;
/

SELECT employee_id, first_name FROM employees WHERE employee_id = 188;

EMPLOYEE_ID FIRST_NAME
----------- ----------
        188 Pat

1 row selected.
```

Deleting XML Data Using DBMS_XMLSTORE

You can use PL/SQL package `DBMS_XMLSTORE` to delete existing data. You specify which rows to delete by calling procedure `setKeyColumn` once for *each* of the columns that are used collectively to identify the row. (In SQL, you would specify the rows using a `WHERE` clause in an `UPDATE` statement.)

Example 12-3 DBMS_XMLSTORE.DELETEXML Example

```
SELECT employee_id FROM employees WHERE employee_id = 188;

EMPLOYEE_ID
-----------
        188

1 row selected.

DECLARE
  delCtx DBMS_XMLSTORE.ctxType;
  rows NUMBER;
  xmlDoc CLOB :=
    '<ROWSET>
       <ROW>
         <EMPLOYEE_ID>188</EMPLOYEE_ID>
         <DEPARTMENT_ID>50</DEPARTMENT_ID>
       </ROW>
     </ROWSET>';
BEGIN
  delCtx  := DBMS_XMLSTORE.newContext('HR.EMPLOYEES');
  DBMS_XMLSTORE.setKeyColumn(delCtx, 'EMPLOYEE_ID');
  rows := DBMS_XMLSTORE.deleteXML(delCtx, xmlDoc);
  DBMS_XMLSTORE.closeContext(delCtx);
END;
/

SELECT employee_id FROM employees WHERE employee_id = 188;
```

```
no rows selected.
```