

XML TO RALATIONAL : BRIDGING THE GAP

Storing XML in traditional storage

Fajar Tjahyono
fajartjahyono@yahoo.com

Lisensi Dokumen:

Copyright © 2003- 2006 IlmuKomputer.Com

Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.

INTRODUCING DBMS_XMLSTORE

The DBMS_XMLSTORE PL/SQL package was introduced in Oracle Database 10g Release 1. This package performs DML operations on relational or object tables inside the Database, based on the contents of an XML document.

Note that before Oracle Database 10g, this functionality existed in another PL/SQL package, called DBMS_XMLSAVE.

CANONICAL XML

In order to use DBMS_XMLSTORE, you need to format your XML documents in Oracle's canonical XML format. "this format is very straightforward; each element in the XML document will map to a column, and the element name will be the column name. Elements that make up a row in the XML document are placed under a <ROW> element, and all of the <ROW> elements are placed inside a <ROWSET> element. If you take two rows from the standard **EMP** table and represent them in Oracle canonical format, you get the following XML document:

```
<ROWSET>
  <ROW>
    <EMPNO>7499</EMPNO>
    <ENAME>ALLEN</ENAME>
    <JOB>SALESMAN</JOB>
    <MGR>7698</MGR>
    <HIREDATE>20-FEB-81</HIREDATE>
    <SAL>1600</SAL>
    <COMM>300</COMM>
    DEPTNO>30</DEPTNO>
  </ROW>
  <ROW>
    <EMPNO>7521</EMPNO>
    <ENAME>WARD</ENAME>
```

```
,JOB>SALESMAN</JOB>
<MGR>7698</MGR>
<HIREDATE>22-FEB-81</HIREDATE>
<SAL>1250</SAL>
<COMM>500</COMM>
<DEPTNO>30</DEPTNO>
</ROW>
</ROWSET>
```

EMPLOYING THE DBMS_XMLSTORE PACKAGE

You can perform the basic **INSERT**, and **UPDATE** operation with **DBMS_XMLSTORE**. The **DBMS_XMLSTORE** package also offers a variety of procedures for customizing these operations. Let's walk through and discuss how to use each of the operations available.

Inserts. **DBMS_XMLSTORE** is the function used to insert rows into the database. In this first example, you create a new table that resembles the **EMP** table called **SALES_EMP** :

```
SQL>create table sales_emp
2 as select * from emp where 1=0
3 /
table created.
```

Now you create an XML document that contains all of the employees from the **SALES** department and insert these employees into the new **SALES_EMP** table. Lines 7 through 18 of listing 1 create the XML document from the **EMP** and **DEPT** tables.

In the anonymous OL/SQL block in listing 1, you can see calls to the **DBMS_XMLSTORE** package to accomplish this insert. Line 21 uses **NEWCONTEXT()** to create a new context using the new table name (**SALES_EMP**). Line 22 calls the **INSERTXML** function, which by default inserts data into every column of the table and returns the number of rows inserted. Finally, output that information using the **DBMS_OUTPUT** package (line 26) and clean up the context in the **DBMS_XMLSTORE** package using the **CLOSECONTEXT()** procedure (line 29).

The following query shows the six rows inserted in **SALES_EMP** by listing 1 :

```
SQL>select empno, ename, job, sal
2      from sales_emp;
```

EMPNO	ENAME	JOB	SAL
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7654	MARTIN	SALESMAN	1250
7698	BLAKE	SALESMAN	2850
7644	TURNER	SALESMAN	1500
7900	JAMES	CLERK	950

code **LISTING 1** : INSERT using DBMS_XMLSTORE

```
SQL>declare
2   l_sales_emp xmltype;
3   l_ctx       dbms_xmlstore.ctxtype;
4   l_rows      pls_integer;
5 begin
6   --get all the sales employees into an xml document
7   select xmlelement("ROWSET",
8     xmlagg(
10      xmlelement("ROW",
11        xmlforest(e.empno,e.ename,e.job,e.mgr,
12          e.hiredate,e.sal, e.comm.,e.deotno)
13      )
14    )
15  )
16  into l_sales_emp
17  from emp e, dept d
18  where d.deptno = e.deptno
19  and d.dname='SALES';
20
21  -- setup our dbms_xmlstore context
22  l_ctx := dbms_xmlstore.newcontext('SALES_EMP');
23  l_rows := dbms_xmlstore.insertxml(
24    l_ctx, l_sales_emp.getClobVal() );
25
26  --how many rows were inserted?
27  Dbms_output.put_line(l_rows || 'rows inserted into SALES_EMP');
28
29  -- clean up
30  dbms_xmlstore.closecontext(l_ctx);
31 end;
31 /
6 rows inserted into SALES_EMP.

PL/SQL procedure successfully completed
```

Updates. Using the example, update a couple of the **SALES_EMP** records to indicate those employees who have received promotions and raises. Here is the XML, that will be used to perform the update :

```
<ROWSET>
<ROW>
  <EMPNO>7499</EMPNO>
  <ENAME>ALLEN</ENAME>
  <JOB>MANAGER</JOB>
  <MGR>7698</MGR>
  <SAL>2600</SAL>
</ROW>
<ROW>
  <EMPNO>7521</EMPNO>
  <ENAME>WARD</ENAME>
  <JOB>MANAGER</JOB>
  <MGR>7698</MGR>
  <SAL>2250</SAL>
```

```
</ROW>
</ROWSET>
```

When performing updates using **DBMS_XMLSTORE**, you need to use other procedures in the package to help the database understand what columns you are updating and what you are using for the update key(s). Listing 2 shows the process for updating the two rows listed above (**ALLEN** and **WARD**).

code **LISTING 2** : UPDATE using DBMS_XMLSTORE

```
SQL>declare
2   l_sales_emp xmltype'
3   l_ctx      dbms_xmlstore.ctxtype;
4   l_rows     pls_integer;
5   begin
6       - simulate the updates to make
7       l_sales_emp :=xmltype('<?xml version="1.0"?>
8       <ROWSET>
9       <ROW>
10          <EMPNO>7499</EMPNO><ENAME>ALLEN</ENAME><JOB>MANAGER</JOB>
11          <MGR>7698</MGR><SAL>2600</SAL>
12          </ROW>
13          <ROW>
14             <EMPNO>7521</EMPNO><ENAME>WARD</ENAME><JOB>MANAGER</JOB>
15             <MGR>7698</MGR><SAL>2250</SAL>
16             </ROW>
17          </ROWSET>');
18
19       - - setup our dbms_xmlstore context
20       l_ctx :=dbms_xmlstore.newcontext('SALES_EMP');
21       A
22       - -setup the columns to be upated
23       dbms_xmlstore.clearupdatecolumnlist(l_ctx);
24       dbms_xmlstore.setupdatecolumn(l_ctx, 'ENAME');
25       dbms_xmlstore.setupdatecolumn(l_ctx, 'JOB');
26       dbms_xmlstore.setupdatecolumn(l_ctx, 'MGR');
27       dbms_xmlstore.setupdatecolumn(l_ctx, 'SAL');
28
29       - -setup the key columns to update by
30       dbms_xmlstorer.setkeycolumn(l_ctx, 'empno');
31
32       - - perform the update
33       l_rows := dbms_xmlstore.updatexml(l_ctx, l_sales_emp.getClobVal( ));
34
35       - -how many rows were updated?
36       Dbms_output.put_line(l_rows || 'rows updated into SALES_EMP');
37
38       - - clean up
39       dbms_xmlstore.closecontext(l_ctx);
40   end;
41 /
42 2 rows updated into SALES_EMP.
43
44 PL/SQL procedure successfully completed
```

STORING XML DATA RELATIONALLY

DBMS_XMLSTORE offers one way to take the content from XML documents and store it in (and remove it from) relational tables. (If you are not yet using Oracle Database 10g, you might try **DBMS_XML** to do the same).