Изменение средствами SQL содержимого таблиц БД по данным XML из внешних источников

Владимир Пржиялковский
Преподаватель технологий Oracle
prz@yandex.ru
open-oracle.ru

Декабрь 2012 г.

Реферат

Приводятся примеры того, как средствами SQL изменять содержимое таблиц БД Oracle на основе данных, представленных в формате XML, и располагающихся в файлах ОС либо в сети.

Введение

При разработке приложения иногда возникает необходимость вносить изменения в обычные таблицы БД по данным, поступающим в формате XML из внешнего источника, такого как файл, на дисковом носителе или же в интернете. Ничего особенного для решения этой задачи не существует; решается она обычными средствами, по общей схеме, с тем лишь отличием, что вместо явного указания текста документа XML требуется сослаться на внешний источник. Обратим внимание на следующее: конструктор XMLTYPE имеет реализации, выполняющие построения объекта этого типа не только по предъявлении документа через параметр XMLDATA типа VARCHAR2, но и (в частности) BFILE. То же самое относится к статической функции CREATEXML типа XMLTYPE.

Тем не менее, некоторые трудности программирования преодолеть придется.

Далее для определенности приводится несколько примеров изменения средствами SQL данных в таблицах БД по внешней информации, поданной в виде XML.

Подготовка к примерам

Для ссылки на содержимое файла ниже будет использован объект БД вида directory:

```
CONNECT / AS SYSDBA

CREATE DIRECTORY extfiles_dir AS 'c:\workdir';

GRANT READ ON DIRECTORY extfiles_dir TO scott;
```

Каталог (директорию) для размещения файлов можно указать по усмотрению. Для начала разместим в этом каталоге файл *depts.xml*, со следующим содержимым:

```
<ROWSET>
   <ROW num="1">
      <DEPTNO>50</DEPTNO>
      <DNAME>MARKETING</DNAME>
      <LOC>MOSCOW</LOC>
   </ROW>
   <ROW num="2">
      <DEPTNO>60</DEPTNO>
      <DNAME>JANITARY</DNAME>
      <LOC>LENINGRAD</LOC>
   </ROW>
```

```
</ROWSET>
```

Вставка строк в таблицу по данным XML из внешнего источника средствами SQL

Рассмотрим задачу добавления строк в таблицу DEPT. Само добавление выполнит команда INSERT, однако надо решить техническую задачу формирования набора строк для неё. С этой целью удобно воспользоваться имеющейся в Oracle функцией XMLTABLE разворачивающей элементы документа XML в столбцы таблицы. Документ XML возьмем из файла *depts.xml*, а с разворачиванием в столбцы придется применить особенный прием.

В первом приближении подготовка данных для вставки могла бы выглядеть так:

```
SELECT *
FROM
   XMLTABLE (
          '/ROWSET/ROW'
          PASSING XMLTYPE ( BFILENAME ( 'EXTFILES_DIR', 'depts.xml' )
                          , NLS_CHARSET_ID ( 'AL32UTF8' ) )
   )
;
```

Полученные таким запросом данные XML развернем в столбцы, добавив в XMLTABLE конструкцию COLUMNS, а во фразу SELECT вставим обращение к этим столбцам:

```
SELECT depts.deptno, depts.dname, depts.loc
FROM
   XMLTABLE (
          '/ROWSET/ROW'
          PASSING XMLTYPE ( BFILENAME ( 'EXTFILES_DIR', 'depts.xml' )
                          , NLS_CHARSET_ID ( 'AL32UTF8' ) )
          COLUMNS deptno NUMBER    ( 2 ) PATH 'DEPTNO'
                , dname  VARCHAR2 ( 14 ) PATH 'DNAME'
                , loc    VARCHAR2 ( 13 ) PATH 'LOC'
   ) AS depts
;
```

Результат:

```
    DEPTNO DNAME          LOC
---------- -------------- -------------
        50 MARKETING      MOSCOW
        60 JANITARY       LENINGRAD
```

После этого сама вставка данных не вызывает затруднений:

```
INSERT INTO dept
   SELECT ... -- как выше
```

Обновление и удаление строк таблицы по данным XML из внешнего источника средствами SQL

Обновление и удаление строк таблицы средствами SQL имеет по сравнению со вставкой дополнительные трудности, связанные с формулировкой оператора DML. В частности, если для удаления строк достаточно воспользоваться оператором DELETE, то для обновления вместо UPDATE придется прибегнуть к оператору MERGE–UPDATE, позволяющему разные строки обновлять по-разному.

Рассмотрим задачу обновления строк таблицы DEPT после осуществленной выше вставки. Данные для обновления хранятся в прежнем каталоге, в файле *deptsupdate.xml* с о следующим содержимым:

```
<ROWSET>
    <ROW>
      <DEPTNO>60</DEPTNO>
```

```
      <LOC>PETROGRAD</LOC>
    </ROW>
    <ROW>
      <DEPTNO>50</DEPTNO>
      <LOC>TVER</LOC>
      <DNAME>DISTRIBUTION</DNAME>
    </ROW>
</ROWSET>
```

Элементом, определяющим строку для изменения, будем полагать `DEPTNO`. Элементы `DEPTNO` сообщают, что изменения должны коснуться отделов с номерами 60 и 50. Техническую трудность составляет определение полей сток, подверженных правке: в нашем случае для отдела 60 это только поле LOC, а для отдела 50 – и LOC, и DNAME. *Для облегчения понимания* избавимся от подзапроса в будущем операторе DML с помощью таблицы с временными данными (в жизни это спорное решение):

```
CREATE GLOBAL TEMPORARY TABLE temp_dept
  AS
  SELECT deptno, dname, loc FROM dept WHERE 1 = 2
;
```

Заполним ее данными из файла *deptsupdate.xml*, подобно тому, как это делалось выше для другого файла:

```
INSERT INTO temp_dept
SELECT depts.deptno, depts.dname, depts.loc
FROM
 XMLTABLE (
    '/ROWSET/ROW'
    PASSING XMLTYPE ( BFILENAME ( 'EXTFILES_DIR', 'deptsupdate.xml' )
                    , NLS_CHARSET_ID ( 'AL32UTF8' ) )
    COLUMNS deptno NUMBER    ( 2 ) PATH 'DEPTNO'
          , dname  VARCHAR2 ( 14 ) PATH 'DNAME'
          , loc    VARCHAR2 ( 13 ) PATH 'LOC'
 ) AS depts
;
```

Выполнение самого обновления с помощью оператора MERGE:

```
MERGE
INTO  dept      d
USING temp_dept t               -- в реальной программе здесь приводим подзапрос
      ON  ( d.deptno = t.deptno )
WHEN MATCHED THEN
     UPDATE SET d.dname = NVL ( t.dname, d.dname ) -- или же  = t.dname
              , d.loc   = NVL ( t.loc,   d.loc   ) -- или же  = t.loc
;
```

Заметьте, что выражения для присвоения составлены из расчета, что если в строке обновляющей таблицы значение не указано, то в строке основной таблице соответствующее значение «не трогается». Если желательно для какого-то столбца интерпретировать отсутствие значения обновления в документе XML как NULL, выражение обновления для него следует сформулировать как D.*столбец* = T.*столбец*. То есть, выбор одного из двух выражений для обновления полностью определяется смыслом, который вложен разработчиками приложения в неуказание элемента в документе XML.

Сформулировать обновление с использованием UPDATE не представляется возможным. Удаление же строк, напротив, не представляется возможным сформулировать через MERGE, и выполняется только через DELETE:

```
DELETE
FROM   dept
WHERE  deptno IN ( SELECT deptno FROM temp_dept )
                  -- в реальной программе приводим основной подзапрос
;
```

Сослаться на исходный документ XML в Сети, вместо файла, позволяет конструктор HTTPURITYPE.

Добавление строк одновременно в несколько таблиц

Наличие в Oracle SQL многотабличной разновидности оператора INSERT позволяет одним действием распределять данные из одного источника по разным таблицам.

Рассмотрим случай, когда в документе XML с новыми данными присутствуют сведения одновременно разных категорий. Например, в следующем документе *newdata.xml* встречаются данные как о сотрудниках, так и об отделах:

```
<ROWSET>
   <employee>
      <DEPTNO>40</DEPTNO>
      <EMPNO>1111</EMPNO>
      <ENAME>Queen</ENAME>
      <SAL>7000</SAL>
   </employee>
   <department>
      <DEPTNO>50</DEPTNO>
      <DNAME>MARKETING</DNAME>
      <LOC>MOSCOW</LOC>
   </department>
   <employee>
      <DEPTNO>40</DEPTNO>
      <EMPNO>2222</EMPNO>
      <ENAME>Duke</ENAME>
      <SAL>4000</SAL>
   </employee>
   <employee>
      <DEPTNO>50</DEPTNO>
      <EMPNO>3333</EMPNO>
      <ENAME>Earl</ENAME>
      <SAL>6000</SAL>
   </employee>
</ROWSET>
```

Считается, что данные должны добавляться в порядке следования: если очередной элемент department, то в таблицу DEPT, а если employee, то в таблицу EMP. Порядок имеет значение: обратите внимание, что сначала предполагается добавить отдел 50, а потом уж в него сотрудника. Отдел 40 уже существует.

Первая задача – построить таблицу, где присутствуют столбцы для всех значений, но по понятным причинам не во всякой строке все значения имеются. Запрос для такой таблицы может выглядеть следующим образом:

```
SELECT
  y.dname
, y.loc
, y.deptno
, y.empno
, y.ename
, y.sal
FROM
  ( SELECT XMLTYPE ( BFILENAME ( 'EXTFILES_DIR', 'newdata.xml')
                   , NLS_CHARSET_ID ( 'AL32UTF8' )
                   )
           empdept
    FROM   dual
  ) xdoc
, XMLTABLE ( 'for $i in /ROWSET/(employee|department) return $i'
             PASSING xdoc.empdept
             COLUMNS
```

```
                dname  VARCHAR2 ( 14 ) PATH '/department/DNAME'
              , loc    VARCHAR2 ( 13 ) PATH '/department/LOC'
              , deptno NUMBER    ( 2 ) PATH './DEPTNO'
              , empno  NUMBER    ( 4 ) PATH '/employee/EMPNO'
              , ename  VARCHAR2 ( 10 ) PATH '/employee/ENAME'
              , sal    NUMBER ( 7, 2 ) PATH '/employee/SAL'
   )
   AS y
;
```

Вот какой получим результат:

```
DNAME       LOC                DEPTNO      EMPNO ENAME             SAL
----------- ------------- ---------- ---------- ---------- ----------
                                   40       1111 Queen            7000
MARKETING   MOSCOW                 50
                                   40       2222 Duke             4000
                                   50       3333 Earl             6000
```

Здесь три строки предназначены предоставить данные о новых сотрудниках, и одна строка о новом отделе.

Полученный таким запросом результат можно использовать для добавления строк одновременно в таблицы EMP и DEPT по следующей схеме:

```
INSERT FIRST
  WHEN empno IS NULL THEN     INTO dept ( deptno, dname, loc )
                              VALUES    ( deptno, dname, loc )
  WHEN empno IS NOT NULL THEN INTO emp ( empno, ename, sal, deptno )
                              VALUES   ( empno, ename, sal, deptno )
SELECT -- ... приводим SELECT как выше
;
```

Упражнение. Подставить в последнюю команду INSERT полное определение запроса SELECT и выполнить добавление строк в таблицы EMP и DEPT.

Приведенный выше пример имеет один скрытый недостаток. В результате запроса SELECT нам требуется иметь вполне определенный порядок строк (если добавляется новый отдел, то сотрудники имеют право добавляться только после него), а у нас отсутствует необходимая для этого фраза ORDER BY. Если Oracle решит вычислять запрос параллельным алгоритмом, порядок может расстроиться, и исполнение INSERT приведет к ошибке. Решение этого вопроса носит технический характер и передается на самостоятельное усмотрение.