Oracle XML DB - Update von XML_Daten

Stephan Karrer

Relationale Welt: Was heißt hier Update?

Aus Sicht der SQL-Welt heisst Update, eine komplette XML-Instanz ersetzt eine vorhandene XML-Instanz

```
CREATE TABLE XML_TABLE OF XMLTYPE;

INSERT INTO XML_TABLE VALUES
     (XMLType('<Termin Nr="498">
                  <Beginn>10.06.03</Beginn>
                  <Ende>13.06.03</Ende>
                </Termin>'));

UPDATE XML_TABLE SET OBJECT_VALUE =
        XMLType('<Termin Nr="498">
                  <Beginn>07.08.17</Beginn>
                  <Ende>09.08.17</Ende>
                 </Termin>') where rownum = 1;
```

XML-Welt: Was heißt hier Update?

Aus Sicht der XML-Welt heisst Update, möglichst effizient einen Teil des XML-Dokuments zu verändern

- Dafür stehen eine Reihe von XML-Funktionen, wie UpdateXML(), zur Verfügung, die allerdings ab Version 12 „Deprecated“ sind
- Stattdessen soll XQUERY-Update-Facility benutzt werden
- Es kann auch ein DOM-API via Paket DBMS_XMLDOM benutzt werden
- Oder wir benutzen Transformation via XSLT oder ...

```
SELECT UPDATEXML(
     XMLType('<Termin Nr="498">
                 <Beginn>10.06.03</Beginn>
                 <Ende>13.06.03</Ende>
                </Termin>'),  -- XML-Instanz
               'Termin/@Nr',  -- Adressierung
               '555')    -- neuer Wert
  as "Result"       FROM dual;
```

Beide Welten zusammen

- Ob ein stückweiser Update einer gespeicherten! XML-Instanz effizient ist, hängt von der Speicherungsform ab
- Die Syntax bleibt stets gleich!

```
-- Ersetzen von 'SBELL' durch 'SKING' via UpdateXML
UPDATE purchaseorder SET OBJECT_VALUE =
  updateXML(OBJECT_VALUE,
            '/PurchaseOrder/User/text()',
            'SKING')
 WHERE XMLExists(
 '$p/PurchaseOrder[Reference="SBELL-20021009123336O1PDT"]'
    PASSING OBJECT_VALUE AS "p");
```

Auswirkung der Speicherungsform

- CLOB
  - Die Instanz wird extrahiert und das DOM-Objekt aufgebaut
  - Via DOM-API wird manipuliert und anschließend die Instanz komplett ersetzt
- Binary XML
  - Hier ist in vielen Fällen kein kompletter DOM-Aufbau nötig
  - Durch das optimierter Speicherformat kann zum betroffenen Stück navigiert und dieses verändert werden
  - Durch die Verwendung von XML-Indizes kann die Navigation verbessert werden
- Objekt-Relational
  - Statt des DOM-Aufbau erfolgt ein XPATH Rewrite (Query Transformation) und es wird im relationalen Sinn nur das entsprechende Stück ersetzt

Statt UPDATEXML mit XQUERY UPDATE

- XQUERY UPDATE ist eine durch das W3C standardisierte Erweiterung von XQUERY
- Wird durch Oracle unterstützt und soll ab 12c Release 1 (12.1.0.1) verwendet werden

```
UPDATE purchaseorder SET OBJECT_VALUE =
  XMLQuery('copy $i := $p1 modify
            (for $j in $i/PurchaseOrder/User
             return replace value of node $j with $p2)
             return $i'
    PASSING OBJECT_VALUE AS "p1",
             'SKING' AS "p2" RETURNING CONTENT)
WHERE XMLExists(
  '$p/PurchaseOrder[Reference="SBELL-20021009123336O1PDT"]'
    PASSING OBJECT_VALUE AS "p");
```

Bisherige XML-Funktionen

- Wurden bis Version 11 üblicherweise benutzt und sind auch weiterhin vorhanden
  - updateXML()
  - insertChildXML()
  - insertChildXMLbefore()
  - insertChildXMLafter
  - insertXMLbefore()
  - insertXMLafter()
  - appendChildXML()
  - deleteXML()
- Verwenden XPATH-Ausdrücke zur Adressierung innerhalb der XML-Instanz

InsertChildXML()

- Fügt neue XML-Knoten in Eltern-Konten ein, die durch den XPATH-Ausdruck adressiert werden

```
CREATE TABLE emp_sample AS
  SELECT employee_id as empid,
         XMLELEMENT("Employee",
                    XMLATTRIBUTES( e.employee_id as "id"),
                    XMLFOREST (e.last_name as "Name",
                               e.salary as "Salary",
                               e.department_id as "Dept_ID")
                    ) AS val FROM employees e;

UPDATE emp_sample e set e.val =
    INSERTCHILDXML(e.val,
                   '/Employee',
                   'Comment',
                   XMLType('<Comment>Kommentar</Comment>'))
WHERE empid =100;
```

InsertChildXML()

- Falls der Zielpfad nichts trifft, wird nichts eingefügt. Das ist kein Fehler!
- Falls mehrere Ziele getroffen werden, wird bei allen eingefügt
- Durch Markieren mit '@' wird ein Attribut eingefügt
- Sofern ein Schema zugrundeliegt, muss die Veränderung natürlich passen (einschließlich Namensraum-Anweisungen)

```
-- Einfuegen eines Attributs
UPDATE emp_sample e set e.val =
                  INSERTCHILDXML(e.val,
                      '/Employee/Salary', '@Unit', 'Euro')
WHERE empid =100;
```

DeleteXML()

- Löscht den adressierten Knoten
- Falls mehrere Ziele getroffen werden, wird bei allen gelöscht
- Das Wurzelelement kann nicht gelöscht werden

```
-- Löschen eines Elements
UPDATE emp_sample e set e.val =
    DELETEXML(e.val, '/Employee/Comment')
WHERE empid =100;

-- Löschen eines Attributs
UPDATE emp_sample e set e.val =
    DELETEXML(e.val, '/Employee/Salary/@Unit')
WHERE empid =100;
```

InsertXMLBefore()

- Fügt einen neuen Wert vor dem adressierten Element (nicht Attribut) ein

```
UPDATE emp_sample set val =
    INSERTXMLBEFORE(val,
                    '/Employee/Dept_ID',
                    XMLType('<Comment>
                              <Author> Karrer </Author>
                              <Date> 10.11.2017 </Date>
                              <Text> Kommentar </Text>
                             </Comment>'))
WHERE empid =100;
```

InsertXMLAfter()

- Fügt einen neuen Wert nach dem adressierten Element (nicht Attribut) ein

```
UPDATE emp_sample set val =
    INSERTXMLAFTER(val,
                   '/Employee/Salary',
                   XMLType('<Comment>
                              <Author> Karrer </Author>
                              <Date> 10.11.2017 </Date>
                              <Text> Kommentar </Text>
                            </Comment>'))
WHERE empid =100;
```

AppendChildXML()

- Fügt einen neuen Wert als letztes Kind unter dem adressierten Element (nicht Attribut) ein

```
UPDATE emp_sample set val =
    AppendChildXML(val,
                   '/Employee/Dept_ID',
                   XMLType('<Comment>
                              <Author> Karrer </Author>
                              <Date> 10.11.2017 </Date>
                              <Text> Kommentar </Text>
                            </Comment>'))
WHERE empid =100;
```

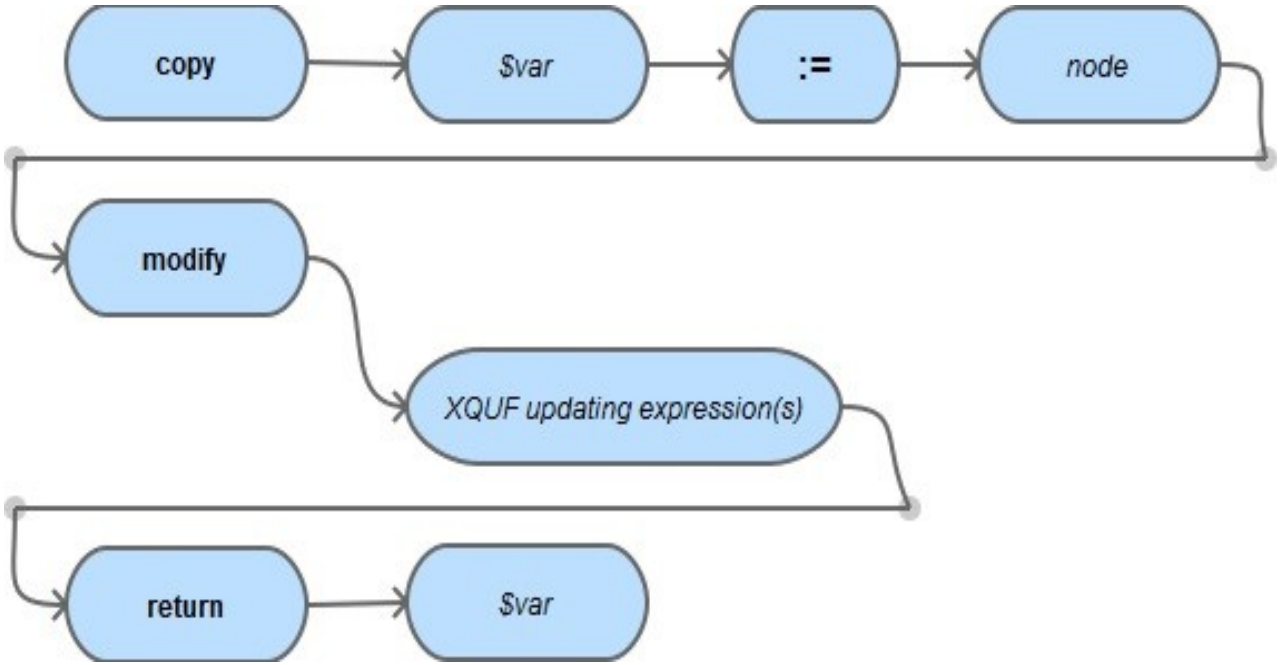
InsertChildXMLBefore(), InsertChildXMLAfter()

- Fügt einen neuen Wert als Kind unter dem adressierten Eltern-Element vor bzw. nach einem angebenen Kindelement ein

```
UPDATE emp_sample set val =
    InsertChildXMLBefore(val,
                         '/Employee',
                         'Dept_ID',
                         XMLType('<Comment>
                                    <Author> Karrer </Author>
                                    <Date> 10.11.2017 </Date>
                                    <Text> Kommentar </Text>
                                  </Comment>'))
WHERE empid =100;
```

XQuery Update Facility

- Bietet Operationen für INSERT, DELETE, REPLACE und RENAME an (XQuery-Ausdrücke)
- Die Anwendung erfolgt bei Oracle unter der Verwendung der TRANSFORM-Operation



Insert mit XQueryUpdate

```
UPDATE emp_sample set val =
    XMLQuery('copy $tmp := . modify insert node
              <Comment>Kommentar</Comment> into
                                          $tmp/Employee
              return $tmp'
              PASSING val
              RETURNING CONTENT)
    WHERE empid = 100;
```

Hinzufügen von Attributen mit XQueryUpdate

- Spezielle Schreibweise ist hierbei erforderlich
  (leider in der Oracle-Dokumentation nicht erläutert)

```
UPDATE emp_sample  set val =
    XMLQuery('copy $tmp := . modify
              insert node attribute Unit{$NEU}
              into $tmp/Employee/Salary
              return $tmp'
          PASSING val ,'Euro' AS neu RETURNING CONTENT)
   WHERE empid = 100;
```

Löschen mit XQueryUpdate

```
-- Löschen eines Elements
UPDATE emp_sample set val =
    XMLQuery('copy $tmp := . modify delete node
              $tmp/Employee/Comment
              return $tmp'
              PASSING val
              RETURNING CONTENT)
    WHERE empid = 100;

-- Löschen eines Attributs
UPDATE emp_sample  set val =
    XMLQuery('copy $tmp := . modify
              delete node $tmp/Employee/Salary/@Unit
              return $tmp'
          PASSING val RETURNING CONTENT)
   WHERE empid = 100;
```

Insert Before mit XQuery

- Schlüsselwort before/after regelt Einfügepunkt

```
UPDATE emp_sample  set val =
    XMLQuery('copy $tmp := . modify
               insert node
                 <Comment>
                   <Author> Karrer </Author>
                   <Date> 10.11.2017 </Date>
                   <Text> Kommentar </Text>

                 </Comment>
               before $tmp/Employee/Dept_ID
               return $tmp'
          PASSING val RETURNING CONTENT)
   WHERE empid = 100;
```

Insert After mit XQuery

- Schlüsselwort before/after regelt Einfügepunkt

```
UPDATE emp_sample  set val =
    XMLQuery('copy $tmp := . modify
                insert node
                  <Comment>
                    <Author> Karrer </Author>
                    <Date> 10.11.2017 </Date>
                    <Text> Kommentar </Text>

                  </Comment>
                after $tmp/Employee/Salary
                return $tmp'
          PASSING val RETURNING CONTENT)
   WHERE empid = 100;
```

Insert zu Beginn/Ende mit XQuery

- Schlüsselwort last/first regelt Einfügepunkt

```
UPDATE emp_sample  set val =
    XMLQuery('copy $tmp := . modify
                insert node
                  <Comment>
                    <Author> Karrer </Author>
                    <Date> 10.11.2017 </Date>
                    <Text> Kommentar </Text>

                  </Comment>
                as last into $tmp/Employee/Dept_ID
                return $tmp'
          PASSING val RETURNING CONTENT)
   WHERE empid = 100;
```

Rename mit XQuery

- Eine einfache Umbenennung gab es bei den bisherigen XML-Funktionen nicht

```
UPDATE emp_sample  set val =
    XMLQuery('copy $tmp := . modify
              rename node $tmp/Employee
              as "Angestellter"
              return $tmp'
         PASSING val RETURNING CONTENT)
   WHERE empid = 100;
```

Ersetzen mit XQuery

- Replace-Operation entspricht im wesentlichen der UpdateXML-Funktion

```
UPDATE emp_sample  set val =
    XMLQuery('copy $tmp := . modify
              replace node $tmp/Employee/Salary
              with <Comment>
                     <Author> Karrer </Author>
                     <Date> 10.11.2017 </Date>
                     <Text> Kommentar </Text>

                   </Comment>
              return $tmp'
         PASSING val RETURNING CONTENT)
   WHERE empid = 100;
```

Ersetzen mit XQuery

- Falls der Pfadausdruck mehr als einen Knoten adressiert, agieren die bisherigen XML-Funktionen auf allen Treffern
- Bei XQuery Update verursacht das Fehler!
  ! Hier ist explizite Iteration via for-Ausdruck notwendig !

```
UPDATE emp_sample set val =
    XMLQuery('copy $tmp := . modify
              ( for $i in  $tmp/Employee/*
                return insert node
                    <Comment>Kommentar</Comment>
                into $i)
              return $tmp'
              PASSING val
              RETURNING CONTENT)
    WHERE empid = 100;
```

Ersetzen von Text-Werten mit XQuery

```
UPDATE emp_sample set val =
    XMLQuery('copy $tmp := . modify
               ( for $i in  $tmp/Employee/*
                 return replace value of node $i
                 with "Ersetzt" )
              return $tmp'
              PASSING val
              RETURNING CONTENT)
    WHERE empid = 100;
```