Przetwarzanie dokumentów XML w Oracle10g: XML DB

Maciej Zakrzewicz
Politechnika Poznańska

Streszczenie

System zarządzania bazą danych Oracle10g posiada szereg wbudowanych mechanizmów, służących do przechowywania, udostępniania, wyszukiwania i przetwarzania dokumentów XML w bazie danych. Najważniejszymi elementami tej zbiorczej funkcjonalności, określanej mianem XML DB, są: typ danych XMLType, umożliwiający umieszczanie dokumentów XML w kolumnach tabel relacyjnych, mechanizmy odwzorowywania dokumentów XML w struktury relacyjne, elementy języka zapytań SQL/XML, parsery dokumentów XML, moduł transformacji XSLT wewnątrz bazy danych, repozytorium dokumentów XML DB Repository. Referat stanowi szczegółowy przegląd funkcjonalności elementów XML DB oraz ich zastosowań.

1 Wprowadzenie

W ostatnich latach gwałtownie wzrosło zapotrzebowanie na przetwarzanie i przechowywanie danych zapisanych w formacie XML. Ponieważ do efektywnego i bezpiecznego przechowywania danych najczęściej wykorzystuje się systemy baz danych, dlatego bardzo aktualnym stało się pytanie o sposoby przechowywania dokumentów XML w systemach relacyjnych i obiektowo-relacyjnych baz danych. Jeżeli na dokument XML spoglądamy jak na zwykły plik tekstowy, wtedy jego przechowywanie w bazie danych nie stwarza żadnych nowych wyzwań (np. tabela z kolumną typu CLOB). Jeżeli jednak chcielibyśmy, aby system zarządzania bazą danych był świadom specyfiki posiadanych danych XML i oferował nam zaawansowane mechanizmy obróbki dokumentów, wtedy niezbędna jest dodatkowa funkcjonalność. Rozszerzenie funkcjonalności systemu zarządzania bazą danych Oracle10g, służące wygodnemu gromadzeniu, generowaniu, przeszukiwaniu i przetwarzaniu dokumentów XML zostało nazwane *XML DB*. Niniejszy artykuł stanowi przegląd własności najważniejszych składników *XML DB*. Omówione zostaną mechanizmy przechowywania dokumentów XML jako wartości typu *XMLType*, możliwości wykorzystywania schematów XML, biblioteki programisty PL/SQL służące do parsowania dokumentów XML, rozszerzenia języka zapytań SQL umożliwiające generowanie dokumentów XML na podstawie treści relacyjnej oraz mechanizm *XML DB Repository*, umożliwiający wykorzystanie bazy danych Oracle10g jako łatwo dostępnej składnicy dokumentów XML.

2 Typ danych XMLType

Do przechowywania i przetwarzania dokumentów XML w obiektowo-relacyjnej bazie danych Oracle10g służy typ danych *XMLType*. Każda wartość typu *XMLType* jest odrębnym, kompletnym dokumentem XML. Fizyczne składowanie dokumentów reprezentowanych przez *XMLType* może odbywać się na dwa sposoby: (1) każdy dokument staje się jednym

dużym obiektem CLOB (domyślnie), (2) każdy dokument jest dekomponowany na rekordy zapisywane w wielu tabelach relacyjnych (wymaga posiadania schematu XML).

Typ *XMLType* może być traktowany tak, jak każdy inny obiektowy typ danych – może być użyty zarówno podczas definicji kolumny w tabeli relacyjnej, jak i podczas tworzenia tabeli obiektów. Poniżej przedstawiono przykład użycia typu danych *XMLType* w tabeli relacyjnej.

```
CREATE TABLE produkty_xml (
  kod NUMBER(11),
  opis XMLType
)

INSERT INTO produkty_xml
VALUES (67653829370, XMLType(
                      '<produkt>
                        <nazwa>Antena dachowa</nazwa>
                        <symbol>1709765</symbol>
                        <cena>85</cena>
                      </produkt>'));

select opis from produkty_xml;

OPIS
-----------------------------------
<produkt>
  <nazwa>Antena dachowa</nazwa>
  <symbol>1709765</symbol>
  <cena>85</cena>
</produkt>
```

Do zaawansowanego przetwarzania dokumentów XML przechowywanych jako *XMLType* służą następujące funkcje języka SQL: *existsNode()*, *extract()*, *extractValue()*, *updateXML()* i *XMLTransform()*. Funkcja *existsNode()* sprawdza, czy wartościowanie podanego wyrażenia XPath względem wartości *XMLType* zwraca co najmniej jeden element lub węzeł tekstowy. Wynikiem działania tej funkcji jest wartość 0 lub 1. Poniżej przedstawiono przykład użycia funkcji *existsNode()*.

```
SELECT kod
FROM produkty_xml
WHERE existsNode(opis, '/produkt[cena=85]') = 1;
```

Funkcja *extract()* zwraca fragment dokumentu XML wskazany wyrażeniem XPath. Funkcja *extractValue()* zwraca wartość skalarną odpowiadającą wynikowi wartościowania wyrażenia XPath. Interesujące działanie realizuje funkcja *updateXML()*, która modyfikuje wybrane węzły w obiekcie *XMLType*. Argumentami funkcji *updateXML()* są pary: wyrażenie XPath, wartość, dzięki czemu możliwe jest modyfikowanie kilku węzłów w ramach jednej operacji. Przykład użycia funkcji *updateXML()* przedstawiono poniżej.

```
UPDATE ksiazki_xml p
SET VALUE(p) = updateXML(VALUE(p), '//cena/text()', '56',
                                   '//rok/text()', 2003)
WHERE extractValue(VALUE(p), '/ksiazka/@isbn')='83-7279-149-X'
```

W celu skrócenia czasu odpowiedzi zapytań dotyczących wartości *XMLType*, użytkownik może skorzystać z indeksów na wyrażeniach (function-based indexes). Przykład tworzenia takiego indeksu przedstawiono poniżej.

```
CREATE INDEX autor_idx ON ksiazki_xml p
(p.extract('//autor').getStringVal())
```

Bardzo interesującą funkcją operującą na wartościach *XMLType* jest *XMLTransform()*. Funkcja ta dokonuje transformacji dokumentu XML w oparciu o dostarczony arkusz stylów XSL. Zarówno dokument źródłowy, jak i arkusz stylów są przekazywane funkcji jako argumenty typu *XMLType*. Wynikiem działania funkcji jest przekształcony dokument XML, reprezentowany przez wartość *XMLType*.

3 Wykorzystywanie schematów XML

System zarządzania bazą danych Oracle10g udostępnia użytkownikom możliwości przechowywania w bazie danych schematów XML (XML Schema), a następnie ich wykorzystywania do: walidacji dokumentów, automatycznego odwzorowywania dokumentów XML w struktury relacyjne, tworzenia tabel lub kolumn *XMLType* opartych na schematach XML oraz poprawy efektywności operacji DML na danych XML. W celu wykorzystywania schematu XML, w pierwszej kolejności użytkownik dokonuje jego rejestracji przy pomocy funkcji *dbms_xmlschema.registerSchema()*. Rejestracja schematu pociąga za sobą szereg niejawnych operacji, wykonywanych przez system zarządzania bazą danych, mających na celu przygotowanie infrastruktury dla składowania i przetwarzania dokumentów XML zgodnych z danym schematem. Operacje te obejmują: utworzenie obiektowych typów danych SQL umożliwiających składowanie dokumentów XML w postaci strukturalnej oraz utworzenie domyślnych tabel dla głównych elementów schematu. Użytkownik posiada możliwość wpływania na odwzorowanie wartości *XMLType* opartych na danym schemacie w struktury relacyjne. Poniżej przedstawiono przykład rejestracji schematu XML, a także wygenerowany typ obiektowy, służący do odwzorowywania dokumentów.

```
DBMS_XMLSCHEMA.registerSchema('ksiazka.xsd',
'<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:xdb="http://xmlns.oracle.com/xdb">
  <xs:element name="ksiazka" xdb:SQLType="KSIAZKA_T">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="tytul" type="xs:string"/>
       <xs:element name="cena" type="xs:float" xdb:SQLName="CENA"
                                               xdb:SQLType="FLOAT"/>
       <xs:element name="autorzy" minOccurs="0" maxOccurs="1"
                                               xdb:SQLType="CLOB">
         <xs:complexType>
           <xs:sequence>
             <xs:element name="autor" type="xs:string"  minOccurs="1"
                         maxOccurs="unbounded"/>
           </xs:sequence>
         </xs:complexType>
       </xs:element>
       <xs:element name="wydawnictwo" type="xs:string"/>
       <xs:element name="rok" type="xs:integer"/>
     </xs:sequence>
     <xs:attribute name="isbn" type="xs:string"/>
   </xs:complexType>
  </xs:element>
</xs:schema>
');
```

```
SQL> desc KSIAZKA_T

 KSIAZKA_T is NOT FINAL
 Nazwa                  Wartość NULL? Typ
 -------------- ------------ ------------------
 SYS_XDBPD$                  XDB.XDB$RAW_LIST_T
 isbn                        VARCHAR2(4000)
 tytul                       VARCHAR2(4000)
 CENA                        FLOAT(126)
 autorzy                     CLOB
 wydawnictwo                 VARCHAR2(4000)
 rok                         NUMBER(38)
```

W oparciu o zarejestrowany schemat, użytkownicy mogą tworzyć tabele, perspektywy lub kolumny typu *XMLType*. Podczas definiowania elementów *XMLType* należy wskazać adres URL, pod którym zarejestrowano schemat XML, oraz wskazać wybrany element główny schematu. Poniżej przedstawiono przykład polecenia tworzącego tabelę relacyjną, zawierającą kolumnę *XMLType* opartą na schemacie XML. Tak utworzona tabela będzie odmawiać przyjmowania dokumentów, które są niezgodne z zadeklarowanym schematem XML.

```
CREATE TABLE ksiazki_col_xsd
(id NUMBER,
 pozycja XMLType
)
XMLTYPE COLUMN pozycja
XMLSCHEMA "ksiazka.xsd" ELEMENT "ksiazka";
```

Interesującym rozwinięciem omawianej funkcjonalności jest możliwość automatycznego generowania schematów XML na podstawie definicji obiektowych typów danych. Służy do tego celu funkcja *dbms_xmlschema.generateSchema()*, która jako argumenty przyjmuje nazwę właściciela i nazwę zdefiniowanego obiektowego typu danych. W wyniku funkcja zwraca wygenerowany schemat XML.

Jeżeli wartości *XMLType* są przechowywane w postaci strukturalnej (relacyjnej), wtedy system zarządzania bazą danych oferuje mechanizm przepisywania zapytań do danych XML. Przepisywanie zapytań polega na konwersji zapytań zawierających wyrażenia XPath do zapytań operujących na tabelach i kolumnach źródłowych. Jest to mechanizm automatyczny, dzięki któremu wykorzystane mogą być klasyczne techniki indeksowania danych relacyjnych, gdyż wyrażenia XPath są przepisywane na proste predykaty selekcji. Poniżej przedstawiono przykład transformacji zapytania skierowanego do tabeli zawierającej dokumenty XML przechowywane strukturalnie.

<u>zapytanie oryginalne:</u>

```
SELECT VALUE(k) FROM katalog k
WHERE extractValue(value(k),'/katalog/ksiazka/rok') = '2002';
```

<u>zapytanie po transformacji:</u>

```
SELECT VALUE(k) FROM katalog k
WHERE k.xmldata.ksiazka.rok = '2002';
```

Nowością w systemie zarządzania bazą danych Oracle10g jest możliwość modyfikacji zarejestrowanego schematu XML przez użytkownika. Za pomocą funkcji *dbms_xmlschema.copyEvolve()*, użytkownik może dokonać zmiany zawartości schematu XML, nie uszkadzając istniejących dokumentów, już o ten schemat opartych.

4 Perspektywy XMLType

Perspektywy *XMLType* są perspektywami budowanymi ponad tabelami obiektowo-relacyjnymi, a zwracającymi obiekty *XMLType*. Umożliwiają one przezroczystą prezentacje istniejących danych w formacie XML. Podczas definiowania perspektyw *XMLType* mogą być wykorzystywane schematy XML. Poniżej przedstawiono przykład polecenia tworzącego i wykorzystującego perspektywę *XMLType*.

```
CREATE VIEW emp_xml_view (doc) AS
SELECT XMLElement(NAME "EMPLOYEE", XMLForest(ename, job, sal))
FROM emp;

SELECT doc FROM emp_xml_view v
WHERE v.doc.existsNode('/EMPLOYEE[SAL>2900]')=1
DOC
------------------------------------------------------------
<EMPLOYEE>
  <ENAME>JONES</ENAME>
  <JOB>MANAGER</JOB>
  <SAL>2975</SAL>
</EMPLOYEE>
<EMPLOYEE>
  <ENAME>SCOTT</ENAME>
  <JOB>ANALYST</JOB>
  <SAL>3000</SAL>
</EMPLOYEE>
...
```

5 Biblioteki programisty dla przetwarzania danych XMLType

Dokumenty XML reprezentowane w postaci wartości *XMLType* mogą być przetwarzane przez programistę PL/SQL przy pomocy biblioteki *PL/SQL Parser API for XMLType* (dbms_xmlparser). Biblioteka ta zawiera parser DOM, służący do transformacji dokumentu XML do standardowej struktury drzewa Document Object Model (DOM). Programista realizuje nawigację wewnątrz drzewa DOM posługując się inną biblioteką: *PL/SQL DOM API for XMLType* (dbms_xmldom). Do transformacji XSL służy kolejna biblioteka, nazywana *PL/SQL XSLT Processor for XMLType* (dbms_xslprocessor). Poniżej przedstawiono przykład anonimowego bloku PL/SQL, przetwarzającego dokument XML w celu wyświetlenia jego wybranych elementów na ekranie.

```
declare
 parser xmlparser.Parser;
 xmlDoc xmldom.DOMDocument;
 titleNode xmldom.DOMNode;
 titleNodeList xmldom.DOMNodeList;
 mydoc XMLType;
begin
 mydoc := XMLType('<katalog><ksiazka><tytul>Zaawansowany ML</tytul></ksiazka>'||
```

```
        '<ksiazka><tytul>SQL/XML</tytul></ksiazka></katalog>');
 parser := xmlparser.newParser;
 xmlparser.parseBuffer(parser, mydoc.getStringVal);
 xmlDoc := xmlparser.getDocument(parser);
 titleNodeList := xmldom.getElementsByTagName(xmlDoc,'tytul');
 for i in 0..xmldom.getLength(titleNodeList) - 1 loop
   titleNode := xmldom.item(titleNodeList, i);
   dbms_output.put_line(xmldom.getNodeValue(xmldom.getFirstChild(titleNode)));
 end loop;
 xmldom.freeDocument(xmlDoc);
 xmlparser.freeParser(parser);
end;
```

W kolejnym przykładzie przedstawiono zastosowanie biblioteki *PL/SQL XSLT Processor for XMLType* do transformacji dokumentu XML znajdującego się poza bazą danych, przy pomocy zewnętrznego arkusza stylów.

```
declare
 parser xmlparser.Parser;  xslProc xslprocessor.Processor;
 xsl xslprocessor.Stylesheet;  xmlDoc xmldom.DOMDocument;
 htmlDoc xmldom.DOMDocumentFragment;
 htmlDocNode xmldom.DOMNode;
 result CLOB;
begin
 dbms_lob.createTemporary(result,true);
 parser := xmlparser.newParser;
 xmlparser.parse(parser,'http://miner/messages.xml');
 xmlDoc := xmlparser.getDocument(parser);
 xslProc := xslprocessor.newProcessor;
 xsl := xslprocessor.newStylesheet('http://miner/messages.xsl',null);
 htmlDoc := xslprocessor.processXSL(xslProc, xsl, xmlDoc);
 htmlDocNode := xmldom.makeNode(htmlDoc);
 xmldom.writeToClob(htmlDocNode, result,'WINDOWS-1250');
 printClobOut(result);
 xmldom.freeDocument(xmlDoc);
 xmlparser.freeParser(parser);
end;
```

6 Rozszerzenia języka zapytań: SQL/XML

SQL/XML stanowi część specyfikacji języka zapytań SQL, opisującą sposoby wykorzystywania SQL w połączeniu z XML. *SQL/XML* rozszerza SQL o funkcje i operatory służące do generowania dokumentów XML na podstawie zawartości relacyjnej bazy danych. Podstawowymi funkcjami *SQL/XML* są: *XMLElement()*, *XMLForest()*, *XMLGen()*, *XMLConcat()*, *XMLAgg()*. Funkcja *XMLElement()* służy do tworzenia elementów XML z danych, które nie są obiektami XML. Poniżej przedstawiono przykład zapytania, generującego dokument XML na podstawie zawartości tabeli *emp*.

```
SELECT e.empno,
       XMLElement ( NAME "emp", e.ename) AS "result"
FROM emp e;

------ ------------------
  7369 <emp>SMITH</emp>
```

```
  7499 <emp>ALLEN</emp>
  7521 <emp>WARD</emp>
  7566 <emp>JONES</emp>
  7654 <emp>MARTIN</emp>
  7698 <emp>BLAKE</emp>
  7782 <emp>CLARK</emp>
  7788 <emp>SCOTT</emp>
  7839 <emp>KING</emp>
  7844 <emp>TURNER</emp>
  7876 <emp>ADAMS</emp>
  7900 <emp>JAMES</emp>
  7902 <emp>FORD</emp>
  7934 <emp>MILLER</emp>
```

Funkcja *XMLForest()* tworzy las elementów XML na podstawie danych nie będących obiektami XML. Podobnie działa funkcja *XMLConcat()*, lecz tworzy ona las elementów poprzez konkatenację elementów XML wywiedzionych z jednego rekordu tabeli. Poniżej przedstawiono przykład zapytania wykorzystującego funkcję *XMLForest()*.

```
SELECT e.empno, XMLForest (e.ename AS "name",
                           e.sal AS "salary"
) AS "result"
FROM emp e
WHERE e.empno < 7600 ;

EMPNO result
----- --------------------
 7369 <name>SMITH</name>
      <salary>800</salary>
 7499 <name>ALLEN</name>
      <salary>1600</salary>
 7521 <name>WARD</name>
      <salary>1250</salary>
 7566 <name>JONES</name>
      <salary>2975</salary>
```

Interesujące jest także działanie funkcji *XMLGen()*, która pozwala generować dokumenty XML na podstawie szablonu realizowanego na źródłowych danych relacyjnych. Poniżej przedstawiono przykład zapytania wykorzystującego funkcję *XMLGen()*.

```
SELECT XMLGen ( '<emp id="{$EMPNO}">
                   <name>{$NAME}</name>
                   <salary>{$SAL}</salary>
                 </emp>',
               e.empno,
               e.ename AS name,
               e.sal
) AS "result"
FROM emp e
WHERE e.empno < 7600 ;

result
-------------------------
<emp id="7369">
  <name>SMITH</name>
  <salary>800</salary>
</emp>
```

```
<emp id="7499">
  <name>ALLEN</name>
  <salary>1600</salary>
</emp>
<emp id="7521">
  <name>WARD</name>
  <salary>1250</salary>
</emp>
<emp id="7566">
  <name>JONES</name>
  <salary>2975</salary>
</emp>
```

Funkcja *XMLAgg()* służy do tworzenia lasu elementów z kolekcji elementów XML wywiedzionych z różnych rekordów tabeli. Poniżej przedstawiono przykład złożonego zapytania, które generuje zbiór zagnieżdżonych elementów XML.

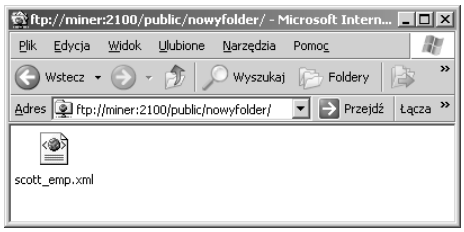
```
SELECT XMLElement ( NAME "department",
         XMLAttributes( e.deptno AS "number" ),
         XMLAgg( XMLElement ( NAME "emp", e.ename )
               )
) AS "result"
FROM emp e
GROUP BY e.deptno ;

result
-------------------------
<department number="10">
  <emp>CLARK</emp>
  <emp>KING</emp>
  <emp>MILLER</emp>
</department>
<department number="20">
  <emp>SMITH</emp>
  <emp>FORD</emp>
  <emp>ADAMS</emp>
  <emp>SCOTT</emp>
  <emp>JONES</emp>
</department>
<department number="30">
  <emp>ALLEN</emp>
  <emp>BLAKE</emp>
  <emp>MARTIN</emp>
  <emp>TURNER</emp>
  <emp>JAMES</emp>
  <emp>WARD</emp>
</department>
```

7 XML DB Repository

Repozytorium dokumentów XML (XML DB Repository) jest usługą systemu zarządzania bazą danych Oracle10g, umożliwiającą przechowywanie i udostępnianie dokumentów XML w formie wirtualnego systemu plików, implementowanego wewnątrz bazy danych. Dostęp do repozytorium dokumentów XML jest możliwy poprzez FTP, WebDAV/HTTP lub SQL. Dokumenty i foldery umieszczane w repozytorium są zapisywane w tabelach bazy danych. Programista może uzyskać dostęp do tych tabel m.in. za pomocą funkcji pakietu *dbms_xdb*.

Poniżej przedstawiono przykład bloku PL/SQL, który tworzy nowy folder oraz umieszcza w nim prosty dokument XML. Dokument ten będzie następnie dostępny np. poprzez FTP.

```
declare
  retb boolean;
begin
  retb := dbms_xdb.createFolder('/public/nowyfolder');
  retb := dbms_xdb.createResource(
    '/public/nowyfolder/scott_emp.xml',
    '<employee><ename>SCOTT</ename></employee>');
  commit;
end;
```



8 Podsumowanie

Rozwiązania zaproponowane w Oracle10g XML DB umożliwiają projektantom baz danych standaryzację stosowanych metod przechowywania dokumentów XML w bazie danych. Dzięki ułatwieniu obsługi danych XML, wielu twórców aplikacji chętniej sięga po ten format podczas realizacji własnych systemów. Niewątpliwie na szczególną uwagę w obrębie całego XML DB zasługują: typ danych *XMLType* oraz rozszerzenie języka zapytań – *SQL/XML*.

9 Referencje

[1] Krzysztof Jankiewicz, Tomasz Traczyk, Marek Wojciechowski, Maciej Zakrzewicz: 'Przechowywanie i przetwarzanie danych XML w systemach baz danych', Materiały II Szkoły PLOUG, Poznań, 2003, ISSN 1641-2117

[2] Oracle XML DB Developer's Guide 10g Release 1, dokumentacja techniczna Oracle