Wykład XI

Aktualizowanie dokumentów XML (Oracle)

do aktualizowania zawartości dokumentów XML służy między innymi funkcja **updateXML**. wynikiem jej działania jest oryginalny dokument ze zmodyfikowanym fragmentem, wyznaczonym przez wyrażenie XPath, podane jako argument. dokument jest aktualizowany jedynie w pamięci, a więc by zmodyfikować zawartość tabeli, konieczne jest wykorzystanie funkcji updateXML wspólnie z poleceniem UPDATE.

składnia:
updateXML(< *obiekt_xmltype* >,<' *wyrażenie_xpath*'>,<*fragment_xml*>)

przykład:
select * from produkty_xml**;**

```
KOD                  OPIS
-------------------- --------------------
125                  <produkt><nazwa>Komputer </nazwa> <cena>1975</cena></produkt>
```

```
UPDATE Produkty_xml  SET  opis = updateXML(opis,
                          '/produkt[nazwa="Komputer "]/cena/text()','1500')
WHERE  kod=125;
```

Oraz inne dostępne w dokumentacji:
http://docs.oracle.com/cd/B28359_01/appdev.111/b28369/xdb04cre.htm#CHDCHHJB

```
insertChildXML
insertChildXMLbefore
insertChildXMLafter
insertXMLbefore
insertXMLafter
appendChildXML
deleteXML
```

Generowanie dokumentów XML z tabel relacyjnych - funkcje SQLX

Funkcja XMLElement i XMLAttributes

- tworzy element XML z danych nie będących XML

Wykład XI

SELECT numer, **XMLElement** (NAME ”pracownik” , nazwisko) wynik
FROM pracownicy ;

```
NUMER WYNIK
---------- --------------------------------------------------
     1000 <pracownik>Lech</pracownik>
     1080 <pracownik>Koliberek</pracownik>
     . . .
     1130 <pracownik>Kolski</pracownik>
     1150 <pracownik>Bulski</pracownik>
```

- atrybuty tworzonego elementu definiowane funkcją **XMLAttributes**

SELECT XMLElement (NAME ”pracownik” , **XMLAttributes** (numer AS “Id”), nazwisko **)** wynik **FROM** pracownicy ;

```
 WYNIK
---------- --------------------------------------------------
<pracownik Id="1000">Lech</pracownik>
<pracownik Id="1080">Koliberek</pracownik>
 . . .
<pracownik Id="1130">Kolski</pracownik>
<pracownik Id="1150">Bulski</pracownik>
```

- można zmienić domyślną nazwę atrybutu podając alias dla kolumny tabeli
- elementy dzieci tworzone są przez zagnieżdżone wywołanie **XMLElement**
- wartość NULL argumentu **XMLElement** powoduje utworzenie pustego elementu

SELECT XMLElement("pracownik", **XMLAttributes**(numer as"id"),
XMLElement(nazwisko, nazwisko), **XMLElement**(pensja,placa+dodatek)) Wynik
FROM pracownicy;

```
WYNIK
--------------------------------------------------------------------------------------------------------------------------
<pracownik id="1000"><NAZWISKO>Lech</NAZWISKO><PENSJA>9810</PENSJA></pracownik>
<pracownik id="1080"><NAZWISKO>Koliberek</NAZWISKO><PENSJA></PENSJA></pracownik>
<pracownik id="1010"><NAZWISKO>Podgajny</NAZWISKO><PENSJA>6180</PENSJA></pracownik>
 . . .
<pracownik id="1150"><NAZWISKO>Bulski</NAZWISKO><PENSJA></PENSJA></pracownik>
```

- **XMLElemen**t może generować elementy o mieszanej zawartości
(elementy - dzieci i tekst)

Wykład XI

SELECT XMLElement(name "szef", **XMLElement**(name "nazwisko",nazwisko),
' szef wszystkich ', **XMLElement**(name "wynagrodzenie",placa+dodatek)) wynik
FROM pracownicy **WHERE** numer=1000;

WYNIK

<szef><nazwisko>Lech</nazwisko> szef wszystkich <wynagrodzenie>9810</wynagrodzenie></szef>

Funkcja XMLForest

- tworzy las elementów XML z danych nie będących XML.
- może być wykorzystana do utworzenia listy elementów - dzieci w elemencie tworzonym funkcją **XMLElement**
- wartość NULL powoduje pominięcie tego elementu

SELECT XMLElement(name "pracownik"**,XMLAttributes(** numer as "Id"**),**
XMLForest(nazwisko, etat, dodatek**))** wynik
FROM pracownicy **;**

WYNIK

<pracownik Id="1000">
<NAZWISKO>Lech</NAZWISKO><ETAT>Dyrektor</ETAT><DODATEK>570</DODATEK>
</pracownik>
<pracownik Id="1080">
<NAZWISKO>Koliberek</NAZWISKO><ETAT>Sekretarka</ETAT>
</pracownik>
<pracownik Id="1010">
<NAZWISKO>Podgajny</NAZWISKO><ETAT>Profesor</ETAT><DODATEK>420</DODATEK>
</pracownik>
. . .
<pracownik Id="1150"><NAZWISKO>Bulski</NAZWISKO><ETAT>Stażysta</ETAT></pracownik>

Funkcja XLMAgg

- tworzy las elementów z kolekcji elementów XML wywiedzionych z różnych wierszy relacji.
- hierarchiczne struktury XML można budować pobierając dane z wielu tabel

SELECT XMLElement(name "Dzial"**,XMLAttributes(** nazwa as "nazwa"**),**
XMLElement(name "liczba" ,count(numer) **),**
XMLAgg(XMLElement(name "pracownik",nazwisko**)**
ORDER BY nazwisko**))** wynik
FROM pracownicy **FULL JOIN** zespoly **USING (**id_zesp**)**
GROUP BY nazwa**;**

Wykład XI

WYNIK
–--
<Dzial nazwa="Administracja"><liczba>2</liczba>
<pracownik>Koliberek</pracownik><pracownik>Lech</pracownik></Dzial>
<Dzial nazwa="Audyt"><liczba>0</liczba>
<pracownik></pracownik></Dzial>
<Dzial nazwa="Bazy danych"><liczba>4</liczba>
<pracownik>Misiecki</pracownik><pracownik>Muszyński</pracownik><pracownik>Podgajny</pracownik>
<pracownik>Rus</pracownik></Dzial>
<Dzial nazwa="Multimedia"><liczba>0</liczba>
<pracownik></pracownik></Dzial>
<Dzial nazwa="Sieci komputerowe"><liczba>4</liczba>
<pracownik>Delecki</pracownik><pracownik>Maleja</pracownik><pracownik>Rajski</pracowni
k><pracownik>Warski</pracownik></Dzial>
<Dzial nazwa="Systemy operacyjne"><liczba>3</liczba>
<pracownik>Kolski</pracownik><pracownik>Lubicz</pracownik><pracownik>Orka</pracownik></Dzial>
<Dzial nazwa="nowy"><liczba>0</liczba><pracownik></pracownik></Dzial>
<Dzial><liczba>1</liczba><pracownik>Bulski</pracownik></Dzial>

PERSPEKTYWY XML

W przypadku udostępniania danych w formacie XML bardzo ważnym mechanizmem stają się perspektywy. Ich zadaniem jest

- *integracja dostępnych informacji o różnej strukturze*
- *umożliwienie nadania ściśle określonej struktury danym semistrukturalnym*

przykład:

CREATE OR REPLACE VIEW Filmoteka_xml(doc)
AS
SELECT XMLElement(name "FILMOTEKA",
XMLAgg(**XMLElement**(name "FILM",**XMLForest**(tytul ,rezyser),**XMLElement**(name "OBSADA",**XMLAgg**(**XMLElement**(name "AKTOR",nazwisko))))))
FROM filmy **FULL JOIN** obsady **USING**(tytul)
GROUP BY tytul,rezyser;

SELECT * FROM Filmoteka_xml;

DOC

<FILMOTEKA>
  <FILM>
        <TYTUL>Dług</TYTUL><REZYSER>Krauze Krzysztof</REZYSER>
              <OBSADA><AKTOR>Chyra</AKTOR></OBSADA>
  </FILM>
  <FILM>
        <TYTUL>Potop</TYTUL><REZYSER>Hoffman Jerzy</REZYSER>
              <OBSADA><AKTOR></AKTOR></OBSADA>
  </FILM>
 . . .
</FILMOTEKA>

Wykład XI

SELECT extract(doc,'//FILM[count(OBSADA/AKTOR)>2]') **FROM** Filmoteka_xml;

WYNIK

```
<FILM>
        <TYTUL>Ogniem i mieczem</TYTUL><REZYSER>Hoffman Jerzy</REZYSER>
                <OBSADA>
<AKTOR>Seweryn</AKTOR><AKTOR>Domogarow</AKTOR><AKTOR>Olbrychski</AKTOR>
                </OBSADA>
</FILM>
```

Pakiet DBMS_XMLGEN

- służy do tworzenia dokumentów XML na bazie wyników zapytań SQL
- utworzone w ten sposób dokumenty stanowią dokładne odzwierciedlenie struktury oraz zawartości tabel, będących wynikami podanych zapytań
- podstawowe funkcje to **getXML( )** i **getXMLType**

p***rzykłady:***

SELECT DBMS_XMLGEN.getxml(' SELECT * FROM filmy **')** wynik **FROM dual;**

WYNIK

```
<?xml version="1.0"?>
<ROWSET>
 <ROW>
  <TYTUL>Powiększenie</TYTUL>
  <REZYSER>Antonioni Michelangelo</REZYSER>
  <KRAJ>GB</KRAJ>
  <ROK>1967</ROK>
  <NAGRODA>T</NAGRODA>
  <KOSZT>3450000</KOSZT>
 </ROW>
  . . .
<ROW>
  <TYTUL>Matnia</TYTUL>
  <REZYSER>Polański Roman</REZYSER>
  <KRAJ>GB</KRAJ>
  <ROK>1966</ROK>
  <NAGRODA>T</NAGRODA>
  <KOSZT>560000</KOSZT>
 </ROW>
</ROWSET>
```

Wykład XI

```
import  java.sql.*;
import  java.io.*;
import  oracle.xml.sql.query.*;
. . .
Connection conn = DriverManager.getConnection
        (”jdbc:oracle:thin:@oracle4.icis.pcz.pl:1521:oracle4”, user , password ) ;
       OracleXMLQuery zap = new OracleXMLQuery( conn,” SELECT * FROM  Pracownicy
       “ );
       zap.setRowTag(“ PRACOWNIK” );
       String xmlString = zap.getXMLString();
       try {
        File outputFile = new File(“ Plik “);
        FileWriter out = new FileWriter ( outputFile );
        out.write (xmlString );
        out.close();
       } catch  ( Exception e ) {
        System.out.println (“ BLEDY”) ;
       }
       conn.close();
. . .
```

```
CREATE  OR  REPLACE  PROCEDURE  PrintClobOut ( result IN OUT NOCOPY CLOB )
IS
xmlstr VARCHAR2(32767);
line VARCHAR2(2000);
BEGIN
  xmlstr:=DBMS_LOB.substr(result,32767);
  LOOP
    EXIT  WHEN xmlstr  IS  NULL;
    line:=substr(xmlstr,1,instr(xmlstr,chr(10))-1);
    DBMS_OUTPUT.put_line('| '||line);
    xmlstr:=substr(xmlstr,instr(xmlstr,chr(10))+1);
  END  LOOP;
END;
```

```
DECLARE
 queryCtx DBMS_XMLQuery.ctxType;
 result CLOB;
BEGIN
 queryCtx := DBMS_XMLQuery.newContext(' SELECT * FROM  Czytelnicy');
 result := DBMS_XMLQuery.getXML(queryCtx);
 printClobOut(result);
DBMS_XMLQuery.closeContext(queryCtx);
END ;
```

Wykład XI

```
| <?xml version = '1.0'?>
| <ROWSET>
|   <ROW num="1">
|     <NUMER>5</NUMER>
|     <NAZWISKO>Domański</NAZWISKO>
|     <IMIE>Tomasz</IMIE>
|     <ADRES>Częstochowa ul Ptasia 4</ADRES>
|     <STATUS>Student</STATUS>
|   </ROW>
 . . .
|   <ROW num="13">
|     <NUMER>29</NUMER>
|     <NAZWISKO>Siedlecka</NAZWISKO>
|     <IMIE>Olga</IMIE>
|     <ADRES>Częstochowa ul Wyzwolenia 33</ADRES>
|     <STATUS>Pracownik</STATUS>
|   </ROW>
| </ROWSET>
```

Język zapytań do XML

powinien:

- umożliwiać zapisanie złożonych warunków wyszukiwania
- być deklaratywny
- zwracać wyniki w tej samej formie, co dane źródłowe
- umożliwiać wykonywanie przekształceń znalezionej informacji, w tym agregacji
- zachować hierarchię i sekwencję struktur

Przegląd języków do xml – trochę historii

- **Lorel** – opracowany w Uniwersytecie Stanforda, składniowo podobny do SQL; był wzorowany na OQL ze standardu ODMG.
- **XML-QL** – opracowano w laboratoriach AT&T. Zapytania składają się z dwóch części where i construct
- **XML-GL** – graficzny język zapytań opracowany na Politechnice Mediolańskiej; jest próbą realizacji języka przyjaznego ludziom na wzór QBE (zapytanie przez przykład)
- **XSLT**- język do definiowania transformacji dokumentów XML w inne dokumenty
- **XQL**- służy do wybierania i filtrowania elementów oraz tekstu z dokumentów XML; nie pozwala na konstruowanie nowych elementów w wyniku zapytania.
- **XPATH**- deklaratywny język wykorzystywany przez kilka innych technologi rekomendowanych przez konsorcjum W3C; między innymi do wyszukiwania wzorców w dokumentach XML.