

# XML Repository

Maciej Zakrzewicz

[mzakrz@cs.put.poznan.pl](mailto:mzakrz@cs.put.poznan.pl)

<http://www.cs.put.poznan.pl/~mzakrz/>

# Charakterystyka XML Repository

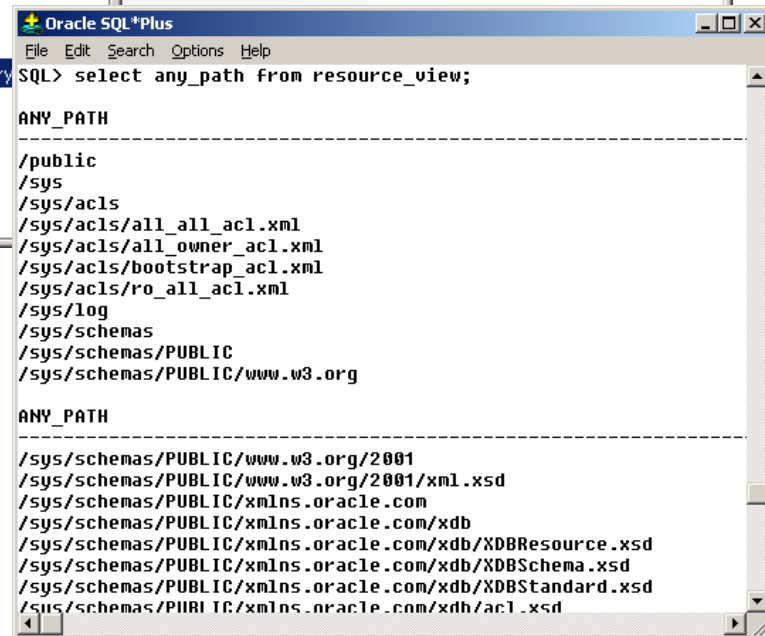
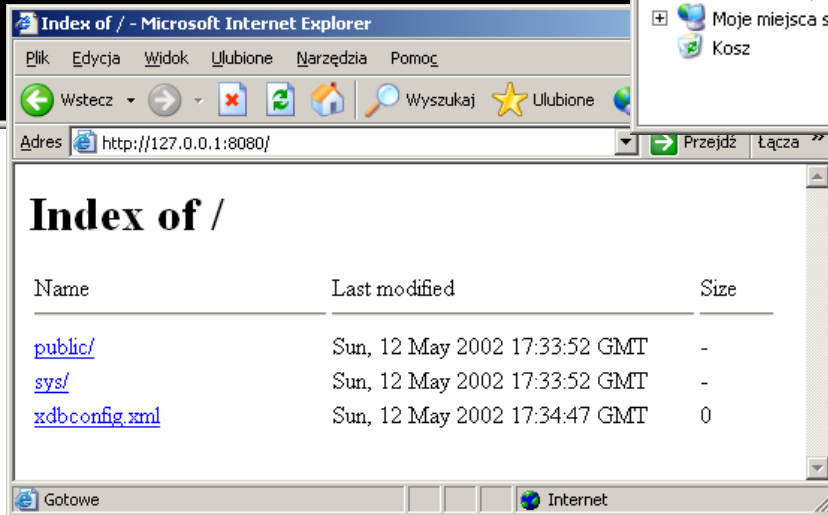
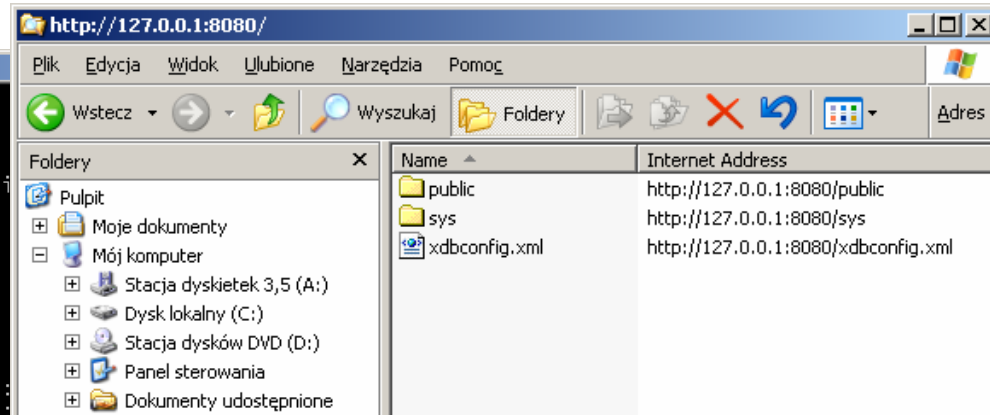
- Repozytorium dokumentów XML (XML Repository) jest usługą serwera bazy danych Oracle9iR2, umożliwiającą przechowywanie i udostępnianie dokumentów XML (również innych plików) w formie wirtualnego systemu plików, implementowanego wewnątrz bazy danych
- Dostęp do repozytorium dokumentów XML jest możliwy poprzez FTP, WebDAV/HTTP lub SQL
- Dokumenty i foldery umieszczane w repozytorium są zapisywane w tabelach bazy danych; programista posiada dostęp do tych tabel m.in. poprzez funkcje pakietu DBMS\_XDB

# Dostęp do XML Repository

WebDAV/HTTP

FTP

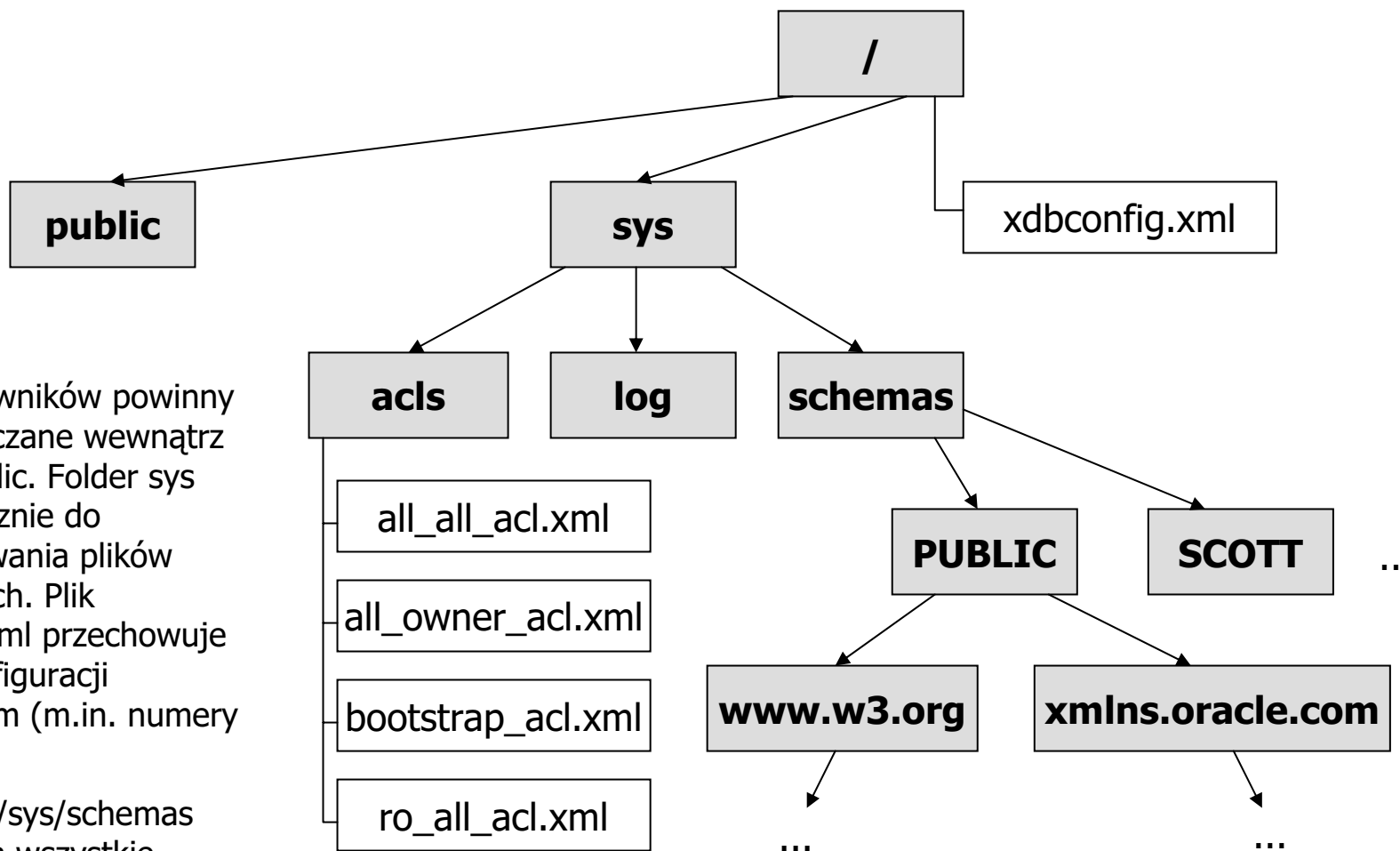
```
C:\WINDOWS\System32\cmd.exe - ftp - ftp
C:\>ftp
ftp> o 127.0.0.1 2100
Połączony z 127.0.0.1.
220 notebook_407 FTP Server (Oracle XML DB/Oracle9i
.2.0.1.0 - Production) ready.
Uzytkownik (127.0.0.1:(none)): scott
331 pass required for SCOTT
Hasło:
230 SCOTT logged in
ftp> dir
200 PORT Command successful
150 ASCII Data Connection
drw-r--r--  2 SYS      oracle      0 MAY 12 17:
drw-r--r--  2 SYS      oracle      0 MAY 12 17:
-rw-r--r--  1 SYS      oracle      0 MAY 12 17:
226 ASCII Transfer Complete
ftp: 190 bajtów odebranych w 0.00sekund 190000.00kb
ftp>
```



HTTP

SQL

# Początkowa struktura folderów



Pliki użytkowników powinny być umieszczane wewnątrz folderu public. Folder sys służy wyłącznie do przechowywania plików systemowych. Plik xdbconfig.xml przechowuje dane o konfiguracji repozytorium (m.in. numery portów).

W folderze /sys/schemas widoczne są wszystkie schematy zarejestrowane w bazie danych.

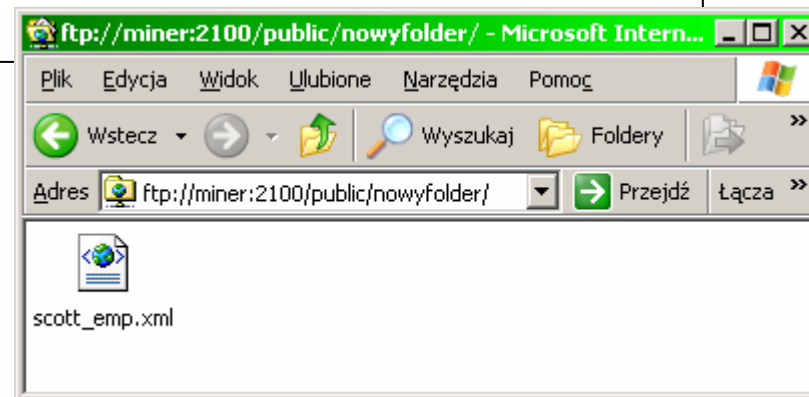
# Pakiet DBMS\_XDB

Funkcja CreateFolder definiuje nowy folder w drzewie dokumentów XML Repository

```
declare
  retb boolean;
begin
  retb := dbms_xdb.createFolder('public/nowyfolder');
  commit;
end;
```

Funkcja CreateResource umieszcza w drzewie dokumentów XML Repository nowy dokument

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



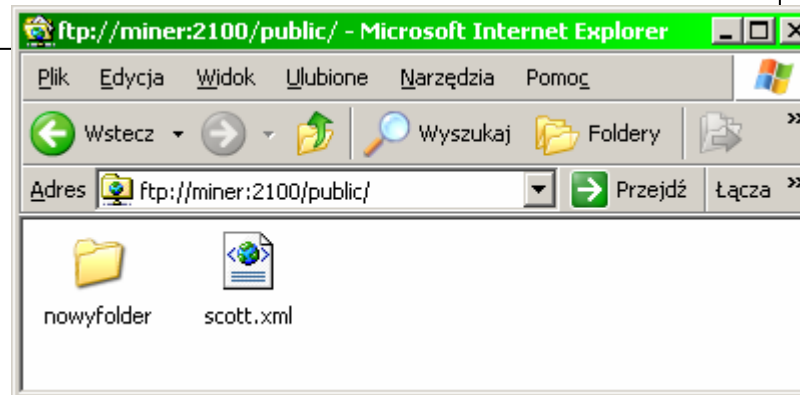
# Pakiet DBMS\_XDB

Funkcja Link definiuje dodatkową ścieżkę (skrót) do dokumentu w XML Repository

```
begin
  dbms_xdb.link('public/nowyfolder/scott_emp.xml','public','scott.xml');
  commit;
end;
```

Funkcja DeleteResource usuwa ścieżkę do dokumentu w XML Repository. Po usunięciu wszystkich ścieżek dokument zostanie usunięty.

```
begin
  dbms_xdb.deleteResource('public/nowyfolder/scott_emp.xml');
  commit;
end;
```



# Dostęp do XML Repository przy pomocy SQL

1. Zarejestruj schemat XML, zgodnie z którym przygotowane będą dokumenty

```
exec dbms_xmlschema.registerURI( 'myemp', 'http://miner/myemp');
```

W bazie danych utworzona zostanie tabela obiektów oraz typy obiektowe, które posłużą do składowania dokumentów wykorzystujących zarejestrowany schemat. Tabela zostanie utworzona na koncie tego użytkownika, który rejestruje schemat XML. Wszystkie dokumenty powołujące się na zarejestrowany schemat będą umieszczane w tej tabeli.

Listę tabel, przechowujących dokumenty XML można uzyskać dzięki zapytaniu do perspektywy słownika danych USER\_XML\_TABLES (lub DBA\_XML\_TABLES)

```
SELECT table_name, xmlschema FROM user_xml_tables;
```

TABLE_NAME	XMLSCHEMA
-----	-----
ROWSET65_TAB	myemp.xsd

2. Korzystając z FTP lub HTTP, umieść w XML Repository dokumenty XML. Każdy dokument powinien powoływać się na zarejestrowany wcześniej schemat. Dokumenty będą automatycznie umieszczane w tabeli powiązanej ze schematem XML.
3. Korzystając z perspektyw RESOURCE\_VIEW, PATH\_VIEW i tabel składowania dokumentów XML, wykonuj zapytania przeszukujące repozytorium XML.

# Perspektywa RESOURCE\_VIEW

Perspektywa RESOURCE\_VIEW zawiera po jednym rekordzie dla każdego pliku przechowywanego w repozytorium. W rekordzie tym zapisana jest ścieżka dostępu oraz obiekt XMLType, reprezentujący metadane dokumentu.

```
SELECT any_path FROM resource_view  
WHERE any_path LIKE '/public%'
```

```
ANY_PATH
```

```
-----  
/public  
/public/nowyfolder  
/public/scott.xml
```

Zawartość dokumentów może zostać odczytana z tabeli, jaka została utworzona podczas rejestrowania schematu XML.

```
SELECT value(p) FROM ROWSET65_TAB;
```

# Oracle XML SQL Utility

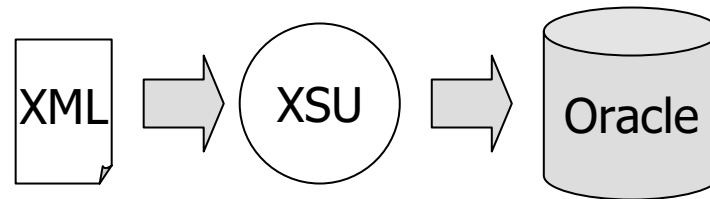
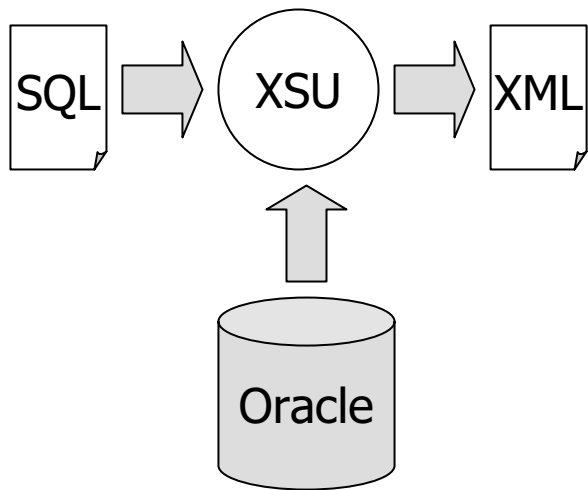
Maciej Zakrzewicz

[mzakrz@cs.put.poznan.pl](mailto:mzakrz@cs.put.poznan.pl)

<http://www.cs.put.poznan.pl/~mzakrz/>

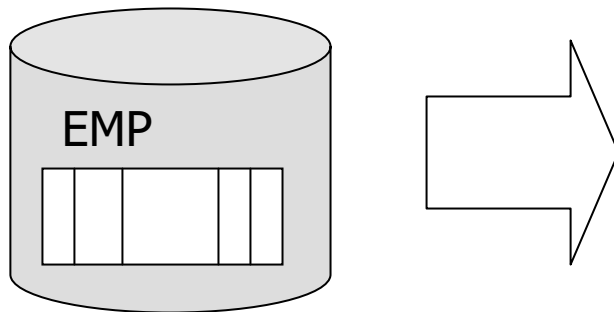
# Oracle XML SQL Utility - XSU

- XML SQL Utility zawiera zbiór klas Java, które umożliwiają realizację następujących zadań:
  - generowanie dokumentów XML na podstawie wyników zapytań SQL
  - ładowanie danych XML do bazy danych Oracle



# Wykorzystanie XSU do generowania dokumentów XML na podstawie zapytań SQL

```
set CLASSPATH=%CLASSPATH%;...\xmlparserv2.jar;...\classes12.jar;...\xsu12.jar
java OracleXML getXML -user scott/tiger "select * from emp"
```



Generowanie dokumentu XML zawierającego wszystkie rekordy tabeli EMP, znajdującej się w bazie danych w schemacie użytkownika SCOTT. Znaczniki rowset i row są generowane automatycznie. Nazwy pozostałych znaczników pochodzą od nazw kolumn tabeli.

```
<?xml version = '1.0'?>
<ROWSET>
  <ROW num="1">
    <EMPNO>7369</EMPNO>
    <ENAME>SMITH</ENAME>
    <JOB>CLERK</JOB>
    <MGR>7902</MGR>
    <HIREDATE>12/17/1980 0:0:0</HIREDATE>
    <SAL>800</SAL>
    <DEPTNO>20</DEPTNO>
  </ROW>
  <ROW num="2">
  </ROW>
</ROWSET>
```

# Wykorzystanie XSU do generowania dokumentów XML na podstawie zapytań SQL

```
java OracleXML getXML -user scott/tiger -withDTD -rowsetTag employees  
-rowTag employee -rowIdColumn EMPNO -useLowerCase -encoding  
windows-1250 "select empno, ename, job, sal, comm from emp"
```

```
<?xml version = '1.0' encoding = 'windows-1250'?>  
<!DOCTYPE employees [  
<!ELEMENT employees (employee)*>  
<!ELEMENT employee (empno, ename?, job?, sal?, comm?)>  
<!ATTLIST employee num CDATA #REQUIRED>  
<!ELEMENT empno (#PCDATA)>  
<!ELEMENT ename (#PCDATA)>  
<!ELEMENT job (#PCDATA)>  
<!ELEMENT sal (#PCDATA)>  
<!ELEMENT comm (#PCDATA)>]>  
<employees>  
  <employee num="7369">  
    <empno>7369</empno>  
    <ename>SMITH</ename>  
    <job>CLERK</job>  
    <sal>800</sal>  
  </employee>  
  ...
```

- withDTD: generuj DTD
- rowsetTag: nazwa znacznika otaczającego
- rowTag: nazwa znacznika rekordów
- rowIdColumn: kolumna generująca identyfikatory
- useLowerCase: znaczniki małymi literami
- encoding: strona kodowa dla znaków

Generowanie dokumentu XML zawierającego wszystkie rekordy tabeli EMP, znajdującej się w bazie danych w schemacie użytkownika SCOTT. Dokument zawiera DTD, znacznik otaczający nazywa się employees, każdy rekord oznaczony jest znacznikiem employee. Wszystkie znaczniki zapisywane są małymi literami. Strona kodowania znaków to windows-1250.

# Wykorzystanie XSU do generowania dokumentów XML na podstawie zapytań SQL

```
java OracleXML getXML -user scott/tiger -withSchema "select ename, job, sal from emp"
```

```
<?xml version = '1.0'?><DOCUMENT xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
  <xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
    <xsd:element name="ROWSET"><xsd:complexType><xsd:sequence>
      <xsd:element name="ROW" minOccurs="0" maxOccurs="unbounded"><xsd:complexType>
        <xsd:sequence>
          <xsd:element name="ENAME" type="xsd:string" nullable="true" minOccurs="0"/>
          <xsd:element name="JOB" type="xsd:string" nullable="true" minOccurs="0"/>
          <xsd:element name="SAL" type="xsd:float" nullable="true" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="num" type="xsd:integer"/>
      </xsd:complexType></xsd:element>
    </xsd:sequence></xsd:complexType></xsd:element>
  </xsd:schema>
<ROWSET xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="#/DOCUMENT/xsd:schema[not(@targetNamespace)]">
  <ROW num="1">
    <ENAME>SMITH</ENAME>
    <JOB>CLERK</JOB>
    <SAL>800</SAL>
  </ROW>
  ...
```

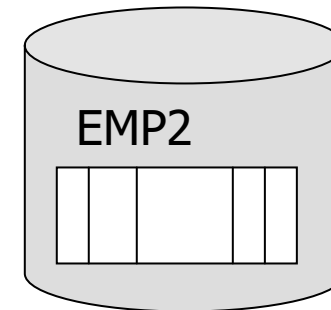
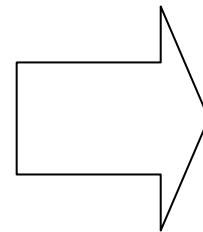
**-withSchema:** generuj  
XML Schema

Generowanie dokumentu XML zawierającego wszystkie rekordy tabeli EMP, znajdującej się w bazie danych w schemacie użytkownika SCOTT. Automatyczne generowanie definicji XML Schema na podstawie definicji tabeli w bazie danych.

# Ładowanie danych XML do tabeli bazy danych przy pomocy XSU

```
java OracleXML putXML -user scott/tiger -filename emp.xml emp2
```

```
<?xml version = '1.0'?>
<ROWSET>
  <ROW num="1">
    <EMPNO>7369</EMPNO>
    <ENAME>SMITH</ENAME>
    <JOB>CLERK</JOB>
    <MGR>7902</MGR>
    <HIREDATE>12/17/1980 0:0:0</HIREDATE>
    <SAL>800</SAL>
    <DEPTNO>20</DEPTNO>
  </ROW>
  <ROW num="2">
    <EMPNO>7499</EMPNO>
    <ENAME>ALLEN</ENAME>
    <JOB>SALESMAN</JOB>
  ...
```



Załadowanie do tabeli EMP2 rekordów zapisanych w pliku emp.xml. Znacznik rowset otacza cały zbiór rekordów, znaczniki row otaczają każdy rekord. Wartości znaczników są ładowane do kolumn o takich samych nazwach, jak nazwy znaczników.

# Wykorzystywanie XSU Java API

```
import oracle.jdbc.driver.*;
import java.sql.*;
import oracle.xml.sql.query.OracleXMLQuery;
...
    DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
    Connection conn = DriverManager.getConnection("jdbc:oracle:oci8:@", "scott", "tiger");
...
OracleXMLQuery qry = new OracleXMLQuery (conn, "select * from emp");
qry.useLowerCaseTagNames();
qry.setRowsetTag("employees");
qry.setRowTag("employee");
qry.setEncoding("windows-1250");
String xmlString = qry.getXMLString();
System.out.println(xmlString);
qry.close();
...
...
XMLDocument xmlDoc = qry.getXMLDOM();
...
```

lub:

Generowanie i wyświetlenie dokumentu XML zawierającego wszystkie rekordy tabeli EMP, znajdującej się w bazie danych w schemacie użytkownika SCOTT.

# Wykorzystywanie XSU Java API

```
import oracle.jdbc.driver.*;
import java.sql.*;
import oracle.xml.sql.dml.OracleXMLSave;
...
    DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
    Connection conn = DriverManager.getConnection("jdbc:oracle:oci8:@", "scott", "tiger");
...
    OracleXMLSave sav = new OracleXMLSave(conn, "emp2");
    sav.insertXML(sav.createURL("file://c:/emp.xml"));
    sav.close();
...
```

Załadowanie do tabeli EMP2 rekordów zapisanych w pliku emp.xml. Znacznik rowset otacza cały zbiór rekordów, znaczniki row otaczają każdy rekord. Wartości znaczników są łaadowane do kolumn o takich samych nazwach, jak nazwy znaczników.

# Wykorzystywanie XSU Java API

```
import oracle.jdbc.driver.*;
import java.sql.*;
import oracle.xml.sql.dml.OracleXMLSave;
...
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection conn = DriverManager.getConnection("jdbc:oracle:oci8:@", "scott", "tiger");
...
OracleXMLSave sav = new OracleXMLSave(conn, "emp");
String [] keyColNames = new String[1];
keyColNames[0] = "EMPNO";
sav.setKeyColumnList(keyColNames);
sav.updateXML(sav.createURL("file://c:/emp.xml"));
sav.close();...
```

Modyfikacja rekordów tabeli EMP. Rekordy wymienione w pliku emp.xml, dopasowywane według wartości kolumny empno, otrzymują nowe wartości tych kolumn, które wymienione są w pliku emp.xml.

```
<?xml version = '1.0'?>
<ROWSET>
  <ROW num="1">
    <EMPNO>7369</EMPNO>
    <ENAME>KOWALSKI</ENAME>
  </ROW>
  <ROW num="2">
    <EMPNO>7499</EMPNO>
    <SAL>2000</SAL>
    <COMM>1500</COMM>
  </ROW>
</ROWSET>
```

# Wykorzystywanie XSU Java API

```
import oracle.jdbc.driver.*;
import java.sql.*;
import oracle.xml.sql.dml.OracleXMLSave;
...
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection conn = DriverManager.getConnection("jdbc:oracle:oci8:@", "scott", "tiger");
...
OracleXMLSave sav = new OracleXMLSave(conn, "emp");
sav.deleteXML(sav.createURL("file://c:/emp.xml"));
sav.close();...
```

Usunięcie z tabeli EMP wszystkich rekordów, które spełniają warunek "empno=7369 and ename='KOWALSKI'" lub "empno=7499 and sal=2000 and comm=1500"

```
<?xml version = '1.0'?>
<ROWSET>
  <ROW num="1">
    <EMPNO>7369</EMPNO>
    <ENAME>KOWALSKI</ENAME>
  </ROW>
  <ROW num="2">
    <EMPNO>7499</EMPNO>
    <SAL>2000</SAL>
    <COMM>1500</COMM>
  </ROW>
</ROWSET>
```

# Wykorzystywanie XSU PL/SQL API

```
declare
  queryCtx DBMS_XMLQuery.ctxType;
  result CLOB;
begin
  queryCtx := DBMS_XMLQuery.newContext('select * from emp');
  result := DBMS_XMLQuery.getXML(queryCtx);
  printClobOut(result);
  DBMS_XMLQuery.closeContext(queryCtx);
end;
```

Generowanie dokumentu XML zawierającego wszystkie rekordy tabeli EMP, znajdującej się w bazie danych. Wynikowy dokument XML znajduje się w zmiennej result, która może zostać zapisana do tabeli, pliku lub wyświetlona na ekranie. Zastąpienie wywołania funkcji getXML() funkcją getDTD() umożliwiłoby wygenerowanie definicji DTD dla konstruowanego dokumentu.

# Przykładowa implementacja procedury PrintClobOut

```
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;
```

Procedura pochodzi z dokumentacji Oracle: Oracle9i XML Developer's Kit Guide -XDK

# Wykorzystywanie XSU PL/SQL API

```
declare
  insCtx DBMS_XMLSave.ctxType;
  rows number;
  xmlDoc CLOB;
begin
  xmlDoc := '<?xml version = "1.0"?><ROWSET><ROW num="1">' ||
            '<EMPNO>7369</EMPNO><ENAME>KOWALSKI</ENAME>' ||
            '</ROW></ROWSET>';
  insCtx := DBMS_XMLSave.newContext('emp');
  rows := DBMS_XMLSave.insertXML(insCtx,xmlDoc);
  DBMS_XMLSave.closeContext(insCtx);
end;
```

Załadowanie podanego dokumentu XML do tabeli EMP. Wstawiony zostanie nowy rekord, w którym wypełnione będą pola empno i ename.

Podobnie realizowane mogą być operacje modyfikacji i usuwania rekordów w oparciu o dokument XML (jak w Java).

# Generowanie dokumentów XML przy użyciu funkcji i pakietów Oracle

Maciej Zakrzewicz

[mzakrz@cs.put.poznan.pl](mailto:mzakrz@cs.put.poznan.pl)

<http://www.cs.put.poznan.pl/~mzakrz/>

# Pakiet DBMS\_XMLGEN

Pakiet DBMS\_XMLGEN stanowi alternatywę dla Oracle XML SQL Utility. Oferuje interfejs PL/SQL i umożliwia generowanie dokumentów XML w oparciu o zapytania SQL.

```
declare
  qryCtx DBMS_XMLGEN.ctxHandle;
  result CLOB;
begin
  qryCtx := DBMS_XMLGEN.newContext('SELECT * FROM emp');
  result := DBMS_XMLGEN.getXML(qryCtx);
  printClobOut(result);
  DBMS_XMLGEN.closeContext(qryCtx);
end;
```

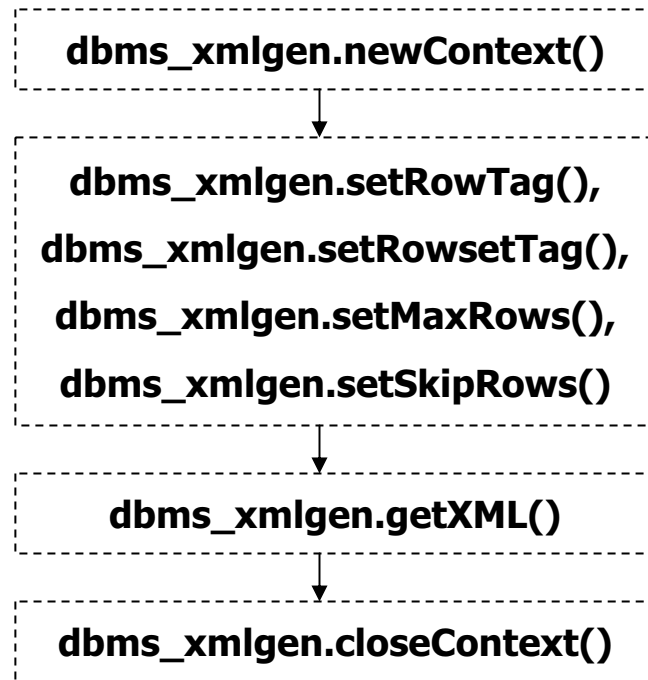
Wynikiem działania funkcji getXML() jest wartość typu CLOB, zawierająca dokument XML.

# Pakiet DBMS\_XMLGEN - transformacja domyślna

- Każdy rekord wyniku zapytania jest zamieniany na znacznik ROW
- Zbiór rekordów wynikowych jest otaczany znacznikiem ROWSET
- Nazwa każdej kolumny wyniku zapytania staje się nazwą znacznika zagnieżdżonego wewnątrz ROW
- Dane binarne są reprezentowane heksadecymalnie

```
<?xml version="1.0"?>
<ROWSET>
  <ROW>
    <EMPNO>7369</EMPNO>
    <ENAME>SMITH</ENAME>
    <JOB>CLERK</JOB>
    <MGR>7902</MGR>
    <HIREDATE>80/12/17</HIREDATE>
    <SAL>800</SAL>
    <DEPTNO>20</DEPTNO>
  </ROW>
  <ROW>
  ...
</ROWSET>
```

# Pakiet DBMS\_XMLGEN - transformacja zaawansowana



**setRowTag():** ustawia nazwę znacznika otaczającego każdy rekord

**setRowsetTag():** ustawia nazwę znacznika otaczającego cały zbiór rekordów

**setMaxRows():** ogranicza liczbę rekordów, na podstawie których powstaje dokument XML w wyniku jednokrotnego wywołania `getXML()`

**setSkipRows():** wskazuje liczbę rekordów, które powinny zostać pominięte, licząc od początku wyniku zapytania

# Pakiet DBMS\_XMLGEN - transformacja zaawansowana

```
declare
  qryCtx DBMS_XMLGEN.ctxHandle;
  result CLOB;
begin
  qryCtx := DBMS_XMLGEN.newContext('SELECT * FROM emp');
  DBMS_XMLGEN.setRowTag(qryCtx, 'EMPLOYEE');
  DBMS_XMLGEN.setRowsetTag(qryCtx, 'EMPLOYEES');
  DBMS_XMLGEN.setMaxRows(qryCtx, 1);
  loop
    result := DBMS_XMLGEN.getXML(qryCtx);
    exit when DBMS_XMLGEN.getNumRowsProcessed(qryCtx) = 0;
    printClobOut(result);
  end loop;
  DBMS_XMLGEN.closeContext(qryCtx);
end;
```

```
<?xml version="1.0"?>
<EMPLOYEES>
  <EMPLOYEE>
    <EMPNO>7369</EMPNO>
    <ENAME>SMITH</ENAME>
    <JOB>CLERK</JOB>
    <MGR>7902</MGR>
    <HIREDATE>80/12/17</HIREDATE>
    <SAL>800</SAL>
    <DEPTNO>20</DEPTNO>
  </EMPLOYEE>
</EMPLOYEES>
<?xml version="1.0"?>
<EMPLOYEES>
  <EMPLOYEE>
    <EMPNO>7499</EMPNO>
```

...

# DBMS\_XMLGEN: transformacja danych obiektowych

```
CREATE TYPE EMP_T AS OBJECT (  
  EMPNO NUMBER(4),  
  ENAME VARCHAR2(10),  
  JOB VARCHAR2(9),  
  SAL NUMBER(7,2)  
);
```

```
CREATE TYPE EMPLIST_T  
AS TABLE OF EMP_T;
```

```
CREATE TYPE DEPT_T AS OBJECT  
(  
  DEPTNO NUMBER,  
  DNAME VARCHAR2(14),  
  LOC VARCHAR2(13),  
  EMPLIST EMPLIST_T  
);
```

```
declare  
  qryCtx DBMS_XMLGEN.ctxHandle;  
  result CLOB;  
  
begin  
  qryCtx := DBMS_XMLGEN.newContext(  
    'SELECT dept_t(deptno,dname,loc,  
      CAST(MULTISET  
        select empno, ename, job, sal from emp e  
        where e.deptno = d.deptno  
      ) AS emplist_t)) AS department  
    FROM dept d);  
  result := DBMS_XMLGEN.getXML(qryCtx);  
  printClobOut(result);  
  DBMS_XMLGEN.closeContext(qryCtx);  
end;
```

```
<?xml version="1.0"?>  
<ROWSET>  
<ROW>  
  <DEPARTMENT>  
    <DEPTNO>10</DEPTNO>  
    <DNAME>ACCOUNTING</DNAME>  
    <LOC>NEW YORK</LOC>  
    <EMPLIST>  
      <EMP_T>  
        <EMPNO>7782</EMPNO>  
        <ENAME>CLARK</ENAME>  
        <JOB>MANAGER</JOB>  
        <SAL>2450</SAL>  
      </EMP_T>  
      <EMP_T>  
        <EMPNO>7839</EMPNO>  
        <ENAME>KING</ENAME>  
        <JOB>PRESIDENT</JOB>  
        <SAL>5000</SAL>  
      </EMP_T>  
      <EMP_T>  
        <EMPNO>7934</EMPNO>
```

...

# Funkcja SYS\_XMLGEN()

Funkcja SYS\_XMLGEN() pobiera wartość skalarną, obiektową lub XMLType i zwraca dobrze uformowany dokument XML reprezentowany przez typ XMLType. W przeciwieństwie do pakietu DBMS\_XMLGEN, który operował na poziomie całego zapytania SQL, funkcja SYS\_XMLGEN() operuje na poziomie pojedynczego rekordu.

```
SQL> SELECT SYS_XMLGEN(ename).getStringVal() FROM emp  
        WHERE ename = 'KING';
```

```
SYS_XMLGEN(ENAME).GETSTRINGVAL()  
-----
```

```
<?xml version="1.0"?>  
<ENAME>KING</ENAME>
```

# Funkcja SYS\_XMLGEN() - parametry

```
SELECT SYS_XMLGEN(ename, XMLFormat.createFormat('name')).getStringVal() FROM emp
```

```
<?xml version="1.0"?>
```

```
<name>SMITH</name>
```

```
<?xml version="1.0"?>
```

```
<name>ALLEN</name>
```

```
<?xml version="1.0"?>
```

```
<name>WARD</name>
```

```
<?xml version="1.0"?>
```

```
<name>JONES</name>
```

```
...
```

Metoda XMLFormat.createFormat() umożliwia wyspecyfikowanie nazwy znacznika otaczającego, nazwę schematu XML, itp.

# SYS\_XMLGEN: transformacja danych obiektowych

```
CREATE TYPE EMP_T AS OBJECT (  
  EMPNO NUMBER(4),  
  ENAME VARCHAR2(10),  
  JOB VARCHAR2(9),  
  SAL NUMBER(7,2)  
);
```

```
CREATE TYPE EMPLIST_T  
AS TABLE OF EMP_T;
```

```
CREATE TYPE DEPT_T AS OBJECT  
(  
  DEPTNO NUMBER,  
  DNAME VARCHAR2(14),  
  LOC VARCHAR2(13),  
  EMPLIST EMPLIST_T  
);
```

```
SELECT SYS_XMLGEN(  
  dept_t(deptno, dname, loc,  
          CAST(MULTISET(  
            select empno, ename, job, sal from emp e  
            where e.deptno = d.deptno  
          ) AS emplist_t))).getStringVal()  
FROM dept d
```

```
<?xml version="1.0"?>  
<ROW>  
<DEPTNO>10</DEPTNO>  
<DNAME>ACCOUNTING</DNAME>  
<LOC>NEW YORK</LOC>  
<EMPLIST>  
<EMP_T>  
<EMPNO>7782</EMPNO>  
<ENAME>CLARK</ENAME>  
<JOB>MANAGER</JOB>  
<SAL>2450</SAL>  
</EMP_T>  
<EMP_T>  
<EMPNO>7839</EMPNO>  
<ENAME>KING</ENAME>  
<JOB>PRESIDENT</JOB>  
<SAL>5000</SAL>  
</EMP_T>  
<EMP_T>  
<EMPNO>7934</EMPNO>  
<ENAME>MILLER</ENAME>  
...
```

# Funkcja SYS\_XMLAGG()

```
SELECT SYS_XMLAGG(SYS_XMLGEN(ename)).getStringVal() FROM emp
```

```
<?xml version="1.0"?>
<ROWSET>
<ENAME>SMITH</ENAME>
<ENAME>ALLEN</ENAME>
<ENAME>WARD</ENAME>
<ENAME>JONES</ENAME>
<ENAME>MARTIN</ENAME>
<ENAME>BLAKE</ENAME>
<ENAME>CLARK</ENAME>
<ENAME>SCOTT</ENAME>
<ENAME>KING</ENAME>
<ENAME>TURNER</ENAME>
<ENAME>ADAMS</ENAME>
<ENAME>JAMES</ENAME>
<ENAME>FORD</ENAME>
<ENAME>MILLER</ENAME>
</ROWSET>
```

Funkcja SYS\_XMLAGG jest funkcją grupową, która łączy wiele dokumentów XML lub ich fragmentów w jeden dokument, otoczony znacznikiem ROWSET (możliwa zmiana przy pomocy XMLFormat)

# Przeszukiwanie danych XML przy użyciu funkcji Oracle Text

Maciej Zakrzewicz

[mzakrz@cs.put.poznan.pl](mailto:mzakrz@cs.put.poznan.pl)

<http://www.cs.put.poznan.pl/~mzakrz/>

# Własności Oracle Text

- Moduł Oracle Text może być wykorzystany do przeszukiwania dokumentów XML przechowywanych w bazie danych
- Podstawową funkcją Oracle Text, znajdującą zastosowanie w przetwarzaniu danych XML jest `Contains()`, służąca do zbadania, czy podane słowo znajduje się we wskazanej sekcji dokumentu
- W celu użycia funkcji `Contains()`, dokumenty XML muszą zostać podzielone na sekcje, a następnie na sekcjach musi zostać utworzony indeks typu `ctxsys.context`
- Podział dokumentu na sekcje może odbywać się automatycznie, w oparciu o wszystkie znaczniki XML, lub w wyniku manualnego deklarowania sekcji przez programistę

# Automatyczny podział dokumentu na sekcje

Utworzenie przykładowej tabeli

```
create table myxml (doc_id number(6), doc xmltype);  
insert into myxml values (1, xmltype.createxml('<ROW num="4"><EMPNO>7566...'));  
...
```

Zbudowanie indeksu na sekcjach automatycznie wygenerowanych ze znaczników

```
create index myindex on myxml(doc) indextype is ctxsys.context  
parameters ('section group ctxsys.auto_section_group');
```

Przeszukiwanie sekcji dokumentu

```
select doc_id from myxml  
where contains(doc, 'salesman within JOB and Allen within ENAME') > 0  
  
select doc_id from myxml  
where contains(doc, '3 within ROW@num') > 0;
```

# Podział na sekcje definiowany przez programistę

Zdefiniowanie wszystkich sekcji w dokumencie (na potrzeby wyszukiwania)

```
begin
  -- wymagane uprawnienie execute na ctx_ddl
  ctx_ddl.create_section_group('myxmlgroup', 'XML_SECTION_GROUP');
  ctx_ddl.add_zone_section('myxmlgroup', 'nazwisko', 'ENAME');
  ctx_ddl.add_attr_section('myxmlgroup', 'num_rekordu', 'ROW@num');
end;
```

Zbudowanie indeksu na zdefiniowanych sekcjach

```
create index myindex on myxml(doc) indextype is ctxsys.context
parameters ('section group myxmlgroup');
```

Przeszukiwanie sekcji dokumentu (tylko tych, które zostały zdefiniowane)

```
select doc_id from myxml
where contains(doc, '3 within num_rekordu') > 0;

select doc_id from myxml
where contains(doc, 'Allen within nazwisko') > 0;
```

# Automatyczny podział na sekcje wg ścieżek

Zastosowanie automatycznego podziału dokumentu na sekcje wg ścieżek

```
begin
  ctx_ddl.create_section_group('xmlpathgroup', 'PATH_SECTION_GROUP');
end;
```

Zbudowanie indeksu na zdefiniowanych sekcjach

```
create index myindex on myxml(doc) indextype is ctxsys.context
parameters ('section group xmlpathgroup');
```

Przeszukiwanie sekcji dokumentu (dostępnych jest wiele elementów języka XPath)

```
select doc_id from myxml
where contains(doc, 'salesman inpath (ROW/JOB)') > 0;

select doc_id from myxml
where contains(doc, 'haspath (//COMM)') > 0;
```