

Generating XML from Relational Tables using ORACLE

by

Selim Mimaroglu

Supervisor: Betty O'Neil

INTRODUCTION

- **Database:** A usually large collection of data, organized specially for rapid search and retrieval
- **Database Management System:** Collection of programs that enables you to store, modify and extract information from a database.
- **Database Schema:** A database schema describes the structure of tables and views in the database.
- **XML:** Extensible Markup Language is a W3C-endorsed standard for document markup. It doesn't have a fixed set of tags and elements.
- **XML Schema:** An XML schema describes the structure of an XML document.

Platform and Technology

- Sun Solaris Operating System 5.8
- Sun Enterprise 250 Server

- Oracle 9.2i
- JAVA
- JDBC (with some Oracle Specific Extensions)
- JSP (Java Server Pages)
- Tomcat 5.0.12 (this is beta version)

What's our task?

- Our task is to generate XML from relational tables
- Input: ITIS (Integrated Taxonomic Information System) Database, XML Schema
- Output: Taxonomic Units in XML format

ITIS Database

- Full database is available at:
http://www.itis.usda.gov/ftp_download.html
- They use Informix
- Includes nonstandard SQL
- Putting this data in Oracle requires some work

Output from Canadian ITIS

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE itis (View Source for full doctype...)>
- <itis>
- <datasource>
  <dbsource>ITIS: Integrated Taxonomic Information System</dbsource>
  <dbserver>ITIS*ca: ITIS Canada (Agriculture & Agri-Food Canada)</dbserver>
  <dbwebaddress>http://sis.agr.gc.ca/itis/</dbwebaddress>
  <dbdatadate>2003-04-11</dbdatadate>
  <dbcurrentdate>2003-11-13</dbcurrentdate>
  <dbexpirydate />
  <dbtermsfuse>This data may be used freely. We only ask that you state that "Taxonomic data is courtesy of the Integrated Taxonomic Information System (http://www.its.usda.gov/index.html)"</dbtermsfuse>
</datasource>
- <taxa>
- <taxon>
  <tsn>601</tsn>
  <concatenatedname>Cyanophycota</concatenatedname>
  <author />
  <url>http://sis.agr.gc.ca/itisca/next?v_tsn=601&p_format=xml&p_lang=</url>
  <rank>Phylum</rank>
  <kingdom>Monera</kingdom>
  <usage>valid</usage>
  <credibilityrating>No review; untreated NODC data</credibilityrating>
- <parent>
  <tsn>202420</tsn>
  <concatenatedname>Monera</concatenatedname>
  <author />
  <url>http://sis.agr.gc.ca/itisca/next?v_tsn=202420&p_format=xml&p_lang=</url>
  <rank>Kingdom</rank>
  <parenttsn>0</parenttsn>
</parent>
- <synonym>
  <tsn>180726</tsn>
```

Output from Canadian ITIS *cont*

```
<synonymname>Cyanophyta</synonymname>
<author />
<rank>Phylum</rank>
<url>http://sis.agr.gc.ca/itisca/next?v\_tsn=180726&p\_format=xml&p\_lang=</url>
</synonym>
- <vernacular>
  <commonname>blue-green algae</commonname>
  <language />
</vernacular>
- <vernacular>
  <commonname>cyanophytes</commonname>
  <language>French</language>
</vernacular>
</taxon>
</taxa>
</itis>
```

Our Output

```
<?xml version="1.0" encoding="iso-8859-1" ?>
- <itis>
- <datasource>
  <dbsource>UMASS: Integrated Taxonomic Information System</dbsource>
  <dbserver>UMASS CS Server</dbserver>
  <dbwebaddress>http://www.cs.umb.edu</dbwebaddress>
  <dbdatadate>2003-04-11</dbdatadate>
  <dbcurrendate>2003-09-25</dbcurrendate>
  <dbexpirydate />
  <dbtermsofuse>This data may be used freely</dbtermsofuse>
</datasource>
- <taxa>
- <taxon>
  <tsn>601</tsn>
  <concatenatedname>Cyanophycota</concatenatedname>
  <url>http://www.cs.umb.edu/v_tsn=601p_format=xml</url>
  <rank>Phylum</rank>
  <kingdom>Monera</kingdom>
  <unit_name1>Cyanophycota</unit_name1>
  <usage>valid</usage>
  <credibilityrating>No review; untreated NODC data</credibilityrating>
  <completeness>unknown</completeness>
  <currency>unknown</currency>
  <initialtimestamp>13-JUN-96 02.51.08.000000 PM</initialtimestamp>
  <taxonupdatedate>1996-07-29</taxonupdatedate>
- <parent>
  <tsn>202420</tsn>
  <concatenatedname>Monera</concatenatedname>
  <url>http://www.cs.umb.edu/v_tsn=202420p_format=xml</url>
  <rank>Kingdom</rank>
  <parenttsn>0</parenttsn>
</parent>
- <vernacular>
```

Our Output *cont*

```
<commonname>blue-green algae</commonname>
<language>unspecified</language>
<approvedind>Y</approvedind>
<vernacularupdatedate>2003-05-09</vernacularupdatedate>
- <othersource>
  <source>NODC Taxonomic Code</source>
  <sourcetype>database</sourcetype>
  <version>8.0</version>
  <acquisitiondate>1996-07-29</acquisitiondate>
  <sourceupdatedate>2003-06-11</sourceupdatedate>
  <originaldescind>U</originaldescind>
  <sourcelinkupdatedate>2003-05-20</sourcelinkupdatedate>
</othersource>
</vernacular>
- <vernacular>
  <commonname>cyanophytes</commonname>
  <language>French</language>
  <approvedind>N</approvedind>
  <vernacularupdatedate>2003-05-21</vernacularupdatedate>
- <othersource>
  <source><a href="http://sis.agr.gc.ca/itis">ITIS*<sup>ca</sup></a></source>
  <sourcetype>website</sourcetype>
  <version>2002</version>
  <acquisitiondate>2003-05-06</acquisitiondate>
  <sourcecomment>\</sourcecomment>
  <sourceupdatedate>2003-05-08</sourceupdatedate>
  <originaldescind>U</originaldescind>
  <sourcelinkupdatedate>2003-05-21</sourcelinkupdatedate>
</othersource>
</vernacular>
- <othersource>
  <source>NODC Taxonomic Code</source>
  <sourcetype>database</sourcetype>
```

Our Output *cont2*

```
<version>8.0</version>
<acquisitiondate>1996-07-29</acquisitiondate>
<sourceupdatedate>2003-06-11</sourceupdatedate>
<originaldescind>U</originaldescind>
<sourcelinkupdatedate>1996-07-29</sourcelinkupdatedate>
</othersource>
- <child>
  <tsn>602</tsn>
  <concatenatedname>Cyanophyceae</concatenatedname>
  <url>http://www.cs.umb.edu/v_tsn=602p_format=xml</url>
  <rank>Class</rank>
</child>
- <child>
  <tsn>610076</tsn>
  <concatenatedname>Prochlorococcus</concatenatedname>
  - <author>
    <taxonauthor>Chisholm et. al., 2001</taxonauthor>
    <authorupdatedate>2002-02-19</authorupdatedate>
  </author>
  <url>http://www.cs.umb.edu/v_tsn=610076p_format=xml</url>
  <rank>Genus</rank>
</child>
</taxon>
</taxa>
</itis>
```

Comparison

- Canadian ITIS doesn't present full data in XML form. We don't know why?
- They obey the XML Schema we have.
- They got synonym relationship wrong.
- You can consider our XML output as an improvement

How?

- There are two methods for creating XML from relational tables using Oracle XMLDB
- Method #1
- Method #2

Don't

- We avoided making any software which gets the data from DBMS and converts it to XML format. Oracle has XML DB Engine built in it.
- We avoided data duplication. We used the database we got from ITIS.

Method #1: Overview

- i. Create Object Types
- ii. Create Nested Tables
- iii. Generate XML from the whole structure

Method #1: a portion from XML Schema

Below is a portion from the XML Schema

```
<xs:element name="comment" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="commentator" type="xs:string" minOccurs="0"/>
      <xs:element name="detail" type="xs:string"/>
      <xs:element name="commenttimestamp" type="xs:string"
minOccurs="0"/>
      <xs:element name="commentupdatedate" type="xs:string"
minOccurs="0"/>
      <xs:element name="commentlinkupdatedate" type="xs:string"
minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Method #1: corresponding Object Type in Oracle

Corresponding Object Type in Oracle would be:

```
create type itcomment as object (  
  commentator varchar2(100),  
  detail char(2000),  
  commenttimestamp timestamp,  
  commentupdatedate date,  
  commentlinkupdatedate date)
```

```
create type itcomment_ntabtyp as table of itcomment  
This matches maxOccurs="unbounded" in the XML  
Schema
```

Method #1: nested tables look like

1	<table border="1"><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>									
2	<table border="1"><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>									
3	<table border="1"><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>									

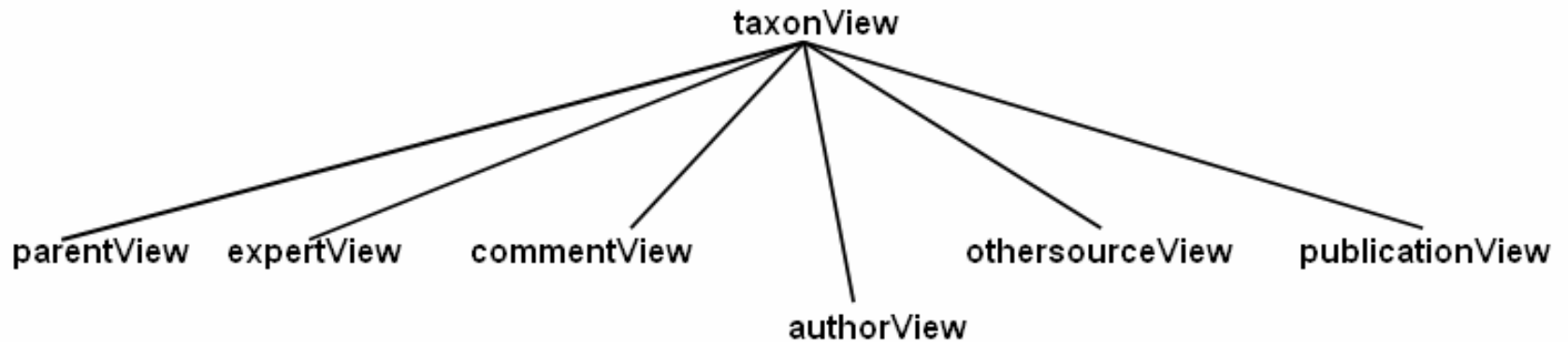
1	<table border="1"><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>									
2	<table border="1"><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>									
3	<table border="1"><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>									

Method #1: XML output (portion)

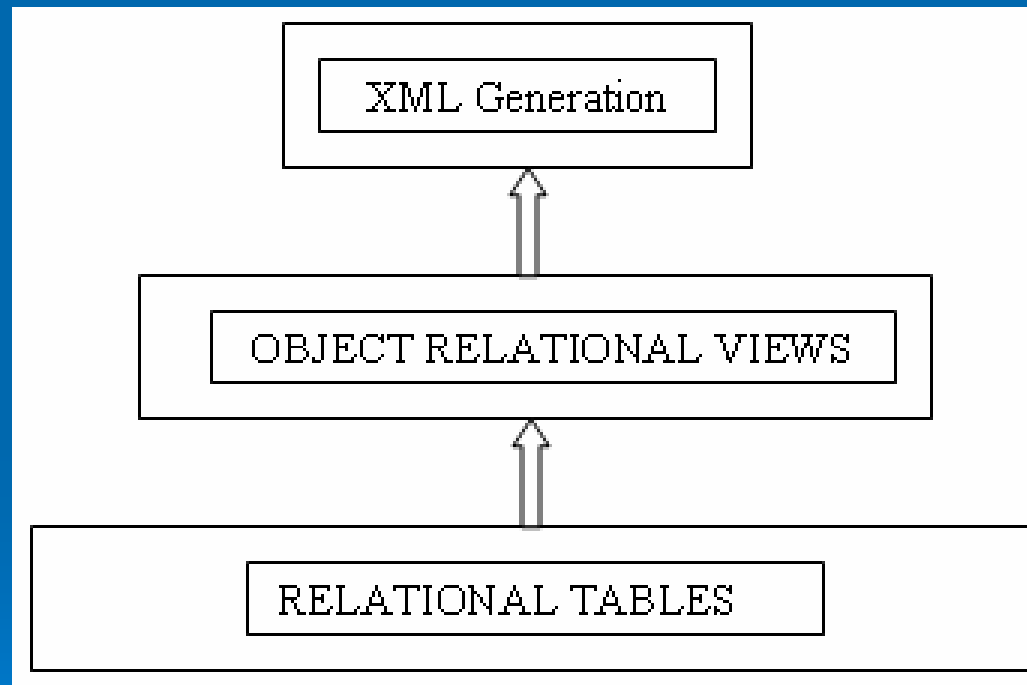
```
.  
.   
<itcomment>  
  <commentator>NODC</commentator>  
  <detail>Part</detail>  
  <commenttimestamp>13-JUN-96 02.51.08.000000  
  PM</commenttimestamp>  
  <commentupdatedate>1996-06-17</commentupdatedate>  
  <commentlinkupdatedate>1996-07-29</commentlinkupdatedate>  
</itcomment>
```

```
.  
.
```

Method #1: view tree



Method #1: Summary



Problems of Method #1

- Redundant Object Relational Layer
- Can't use keywords for naming the elements. For example I had to create *itcomment* Object Type instead of *comment*.

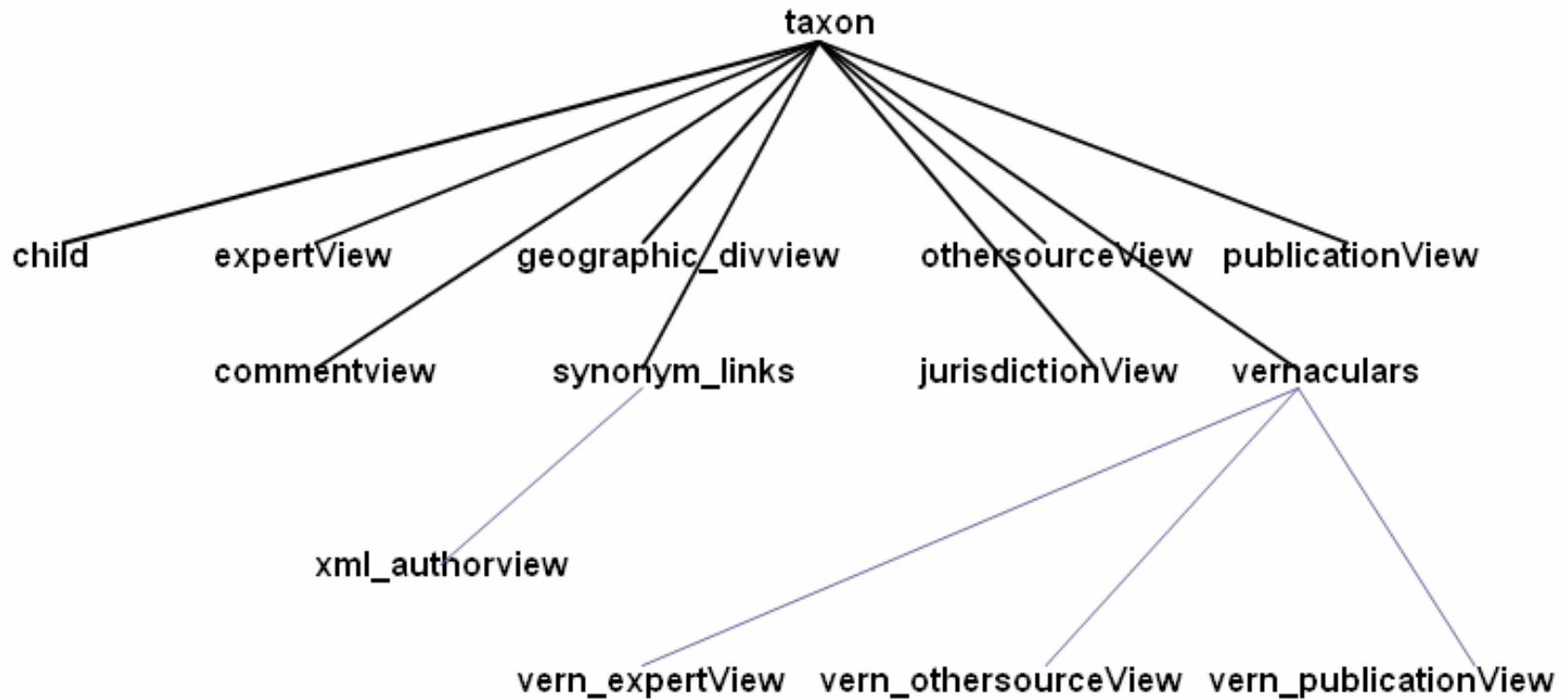
Method #2: Overview

- In this approach, bypass Object Relational Representation
- Create XML from Relational Tables directly
- Two levels only
- XMLType View
- Only one SQL query in view definition which creates everything we need

Method #2: XML generation

```
CREATE or REPLACE VIEW txn of XMLTYPE with object id
(extract(sys_nc_rowinfo$, '/taxon/tsn/text()').getnumberval())
AS SELECT xmlelement("taxon",
    xmlforest(
        t.tsn as "tsn",
        --trim(t.unit_name1) || ' ' || trim(t.unit_name2) as "concatenatedname",
        (SELECT con_name(t.tsn) from dual) as "concatenatedname",
        (SELECT xmlforest(av.taxon_author as "taxonauthor",
            to_char(av.update_date, 'YYYY-MM-DD')
            "authorupdatedate")
        FROM xml_authorview av
        WHERE t.tsn = av.tsn) "author",
        'http://www.cs.umb.edu/v_tsn='||t.tsn||'p_format=xml' as "url",
        (select rank_name(t.rank_id) from dual) as "rank",
        (select
            trim(k.kingdom_name)
            .
            .
            .
        )from taxonomic_units t where t.tsn=1100 ;
```

Method #2: view tree



SQLX Functions

- SQLX standard, an emerging SQL standard for XML.
- XMLElement()
- XMLForest()
- XMLConcat()
- XMLAgg()
- Because these are emerging standards the syntax and semantics of these functions are subject to change in the future in order to conform to the standard.

Using SQLX Functions

- **XMLElement()** Function: It takes an element name, an optional collection of attributes for the element, and zero or more arguments that make up the element's content and returns an instance of type `XMLType`.
- **XMLForest()** Function: It produces a forest of XML elements from the given list of arguments

Using SQLX Functions *cont*

- XMLAgg() Function: It is an aggregate function that produces a forest of XML elements from a collection of XML elements.
- XMLConcat() Function: It concatenates all the arguments passed in to create a XML fragment.

We don't own the data!

- It's pure relational
- It makes difference
- If our database were in Object relational form, we would have used Method #1
- Not a good idea, to change the paradigm.

We generate XML on fly

- Dynamic view
- We don't generate XML for the whole database and get a portion of it
- We generate what we need only

How does Oracle store XML

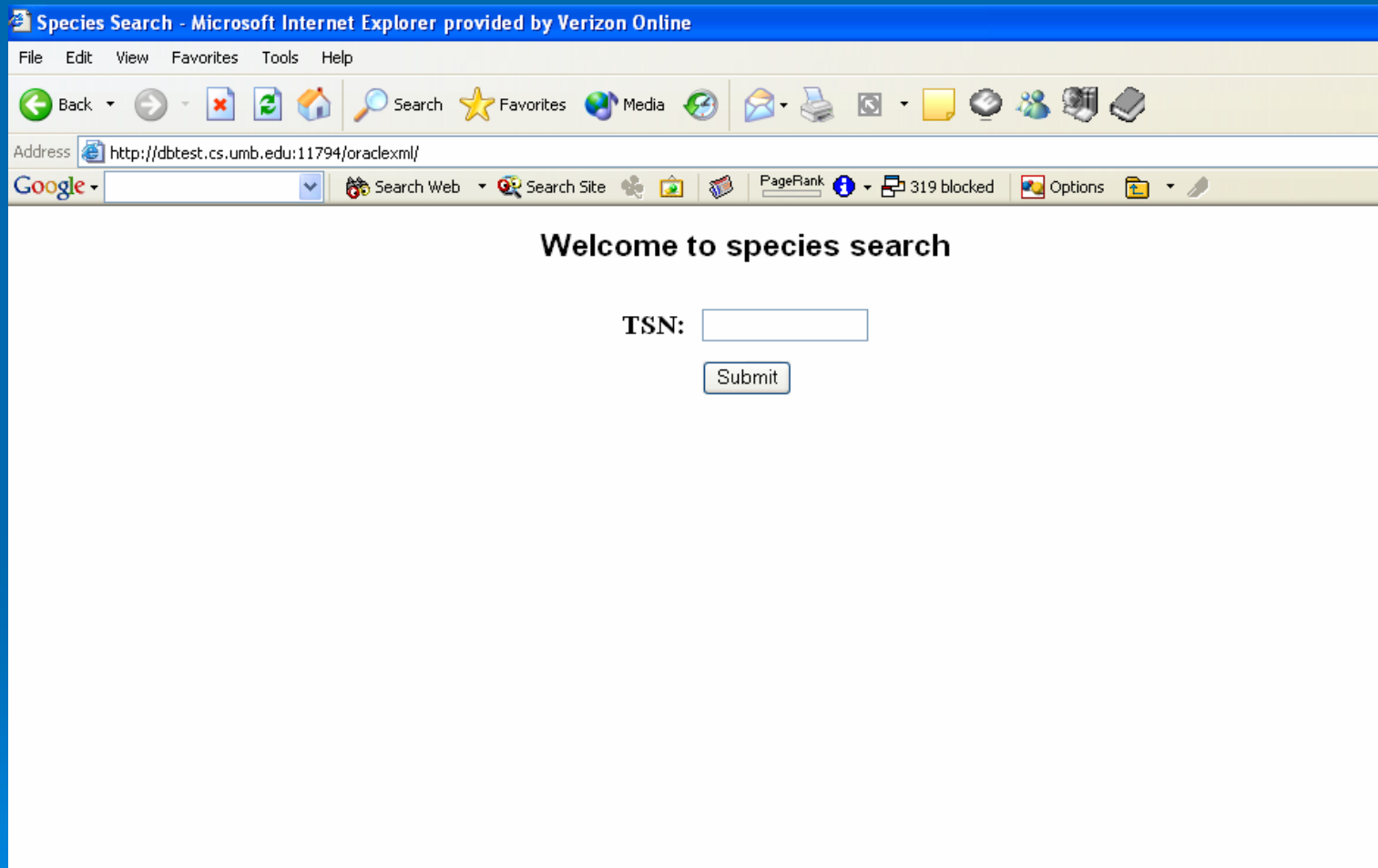
- In underlying object type columns. This is the default storage mechanism.
- In a single underlying LOB column. Here the storage choice is specified in the **STORE AS** clause of the **CREATE TABLE** statement:

```
CREATE TABLE po_tab OF xmltype  
STORE AS CLOB
```

Relation Between XMLSchema and SQL Object-Relational Types

- When an XML schema is registered, Oracle XML DB creates the appropriate SQL object types that enable structured storage of XML documents that conform to this XML schema. All SQL object types are created based on the current registered XML schema, by default.
- It's possible to do this manually, that's what we did in Method #1

XML Delivery



XML Delivery *cont*

```
http://dbtest.cs.umb.edu:11794/oraclexml/query.jsp - Microsoft Internet Explorer provided by Verizon Online
File Edit View Favorites Tools Help
Back Forward Stop Home Search Favorites Media RSS Print Mail News Groups
Address http://dbtest.cs.umb.edu:11794/oraclexml/query.jsp
Google Search Web Search Site PageRank 319 blocked Options
<?xml version="1.0" encoding="iso-8859-1" ?>
- <itis>
- <datasource>
  <dbsource>UMASS: Integrated Taxonomic Information System</dbsource>
  <dbserver>UMASS CS Server</dbserver>
  <dbwebaddress>http://www.cs.umb.edu</dbwebaddress>
  <dbdatadate>2003-04-11</dbdatadate>
  <dbcurentdate>2003-09-25</dbcurentdate>
  <dbexpirydate />
  <dbtermsofuse>This data may be used freely</dbtermsofuse>
</datasource>
- <taxa>
- <taxon>
  <tsn>605</tsn>
  <concatenatedname>Agmenellum</concatenatedname>
- <author>
  <taxonauthor>De Brebisson, 1839</taxonauthor>
  <authorupdatedate>1996-07-29</authorupdatedate>
</author>
  <url>http://www.cs.umb.edu/v_tsn=605p_format=xml</url>
  <rank>Genus</rank>
  <kingdom>Monera</kingdom>
  <unit_name1>Agmenellum</unit_name1>
  <usage>valid</usage>
  <credibilityrating>No review; untreated NODC data</credibilityrating>
  <completeness>unknown</completeness>
```

XML Delivery *cont2*

- Java Server Page get request from the user, passes it to Java Bean
- Java Bean connects Oracle using JDBC, gets information, passes it to JSP
- Used some nonstandard features of Oracle JDBC

XML Delivery Performance

- XML output is (very) fast in the database: 0.032 seconds average
- Use “thin” instead of “oci” driver
- Use Connection Pool
- Use OracleStatement instead of OraclePreparedStatement
- Turn off auto-commit feature
- Define column type
- Better performance maybe possible with Java Stored Procedures in Oracle. We will try and find out

Conclusion

- Although we have implementation both Methods; #1 and #2, we chose to use Method #2: Using XMLType View.
- It's fast and easy
- Using O-R approach makes sense when you have your data in O-R form
- Mostly I found Oracle documentation useful. At some places it was fuzzy and hard to understand. This is recent feature addition to Oracle. We see brand new docs.

Questions?

Thanks

- Send comments, suggestions to smimarog@cs.umb.edu
- You can find this presentation at <http://www.cs.umb.edu/~smimarog/talks/xmlTalk.ppt>