CONTENT-PUBLISHING APPLICATION DEVELOPMENT USING ORACLE 10g, JAVA AND XSLT

Daniela Nikolić¹, Dražen Nikolić²
¹Univerzitet u Nišu
²ActiveMedia, Niš

I INTRODUCTION

As companies increasingly depend on the Internet to conduct business, it is no longer satisfactory to run applications that depend upon paper to exchange data. These applications must be able to exchange data without human intervention and at the speed of the Internet. Each of these business areas has a set of requirements in these application areas which can be met through the family of XML technologies.

When data is incorporated along with its metadata in XML, it is more self-describing and portable and can be easily shared, transformed and transported across applications and platforms. This has been the driving force in making XML a widely accepted format for encapsulating data and in a standard protocol for delivering services across software components in business systems.

As a result, the ways of managing XML data greatly influence the architecture of modern databases. Oracle Database (DB) 10g provides native storage, retrieval, and processing of XML. This native XML functionality is integrated with the Oracle relational database server to bridge the gap between the relational table-row and XML hierarchical storage. Therefore, users have the advantage of using the power of both.

The overall XML infrastructure in Oracle DB provides both high-performance native XML support and an extensible platform on which users can build and deploy their own solutions.

The purpose of this paper is to describe the way of developing Web content-publishing application using Oracle DB 10g, Java and XSLT.

II RELATIONAL TABLES AND XML DATA

Relational tables are normally designed without considering XML storage. However, in many cases these tables can be used to store “shredded” XML documents and produce a useful XML representation.

For a data-centric XML document, if XML elements and attributes are properly stored in relational table columns, it has the optimized storage for further processing, which avoids the overhead of keeping and managing the XML structure in the database.

Using relational tables for storing XML data offers many advantages:

- Relational modeling with data normalization
- Relational storage that handles the schema evolution well
- No XML overhead after data is stored in database
- Easy reuse of the data
- High performance for SQL queries
- No DOM fidelity
- Full support for data replication.

On the other hand, disadvantages are:

- Low throughput for data uploading
- Table joins are required for XML queries.

Oracle DB 10g provides extensive supports for loading, exporting, and processing XML data into relational tables.

For loading XML data in Oracle DB 10g following utilities can be used:

- XML SQL Utility (XSU)
- TransX Utility
- DBMS_XMLSTORE package.

If the functionality provided by these utilities is not sufficient, their programmatic APIs in conjunction with other XDK (XML Developer’s Kit) libraries can be used for building custom solutions.

To generate XML from the SQL data stored in object or relational tables, Oracle DB 10g offers:

- SQL/XML functions
- DBMS_XMLGEN package
- XMLType views.

By using Oracle SQL extensions and the SQL/XML functions for XML processing, relational data model and XML data model can be easily used in one SQL query. This gives users the maximum flexibility for solving business problems [1].
### III XSLT

XSLT stands for the XML Stylesheet Language for Transformations. There are actually dual purposes for the XSLT technology as follows:

- Transform XML documents into other XML documents
- Transform XML input into an output format appropriate for a target device.

The first usage of XSLT is relevant to data exchange. XSLT can be used to extract interesting subsets of XML documents and make them into XML documents. Or it can convert a document conforming to certain encoding rules into one that is similar but conforms to slightly different rules. In this case XSLT becomes a kind of adapter for converting between representational models.

The second of these is mostly used when a single content source (document of some type) needs to be rendered in a number of different formats, perhaps using different markup conventions. Rather than create multiple versions of the content (one for each target format), XSLT allows creating different transforms of the content into each of the target rendering formats [2].

So, data in XML format can be easily transformed using XSLT to various presentation formats, such as HTML, WML (Wireless Markup Language), SVG, or any other Web publishing format that clients request. Because of that, XML is widely used in content management and Web publishing systems. These kinds of applications normally have the structure shown in Figure 1:

### IV SYSTEM ARCHITECTURE

The flight-booking application presented in this paper is designed as a web application running on a web servlet engine, and it is used for searching for available flights.

For developing the application the following was used:

1. Jakarta Tomcat 4.0.1 as Web servlet engine
2. Oracle DB 10g for data storage
3. Java version 1.4.1 for writing business logic
4. JDBC thin driver for Oracle 10g for connecting to database
5. MSIE 6.0 as desktop computer test browser
6. Nokia Mobile Browser 4.0 as mobile test browser simulator

### V APPLICATION WORKFLOW

Java servlet (object which extends javax.servlet.http.HttpServlet class) takes the request from a client browser with the flights search criteria fields (as request parameters) entered from a flights search form. Each client browser type has its own look & feel search form (depending of the screen measurement), but the request parameter names are the same on all search forms. Using java.sql.CallableStatement the servlet than sends input parameters to Oracle database and executes stored function in order to search over available flights which satisfy given search criteria.

Communication between servlet engine and Oracle database is performed by JDBC thin driver.

The results from Oracle stored function is CLOB object presenting XML data with found flights. For generating XML data the following cursor was used:

```sql
cursor curXML is
  select XMLForest(F.ID as id, 
    class_type(F.CLASS, C.NAME) as class, 
    to_char(F.DEPARTURE_DATE_TIME, 'dd.mm.yyyy hh24:mi') as departure_date_time, 
    to_char(F.ARIVE_DATE_TIME, 'dd.mm.yyyy hh24:mi') as arrive_date_time, 
    airport_type(F.DEPARTURE_AIRPORT, AD.KEY, AD.NAME) as departure_airport, 
    airport_type(F.ARRIVE_AIRPORT, AA.KEY, AA.NAME) as arrive_airport, 
    F.PRICE_ADULT, 
    F.PRICE_CHILDREN, 
    F.PRICE_INFANT) as XML_COL 
  from FLIGHT F, 
    CLASS C, 
    AIRPORT AD, 
    AIRPORT AA 
  where F.DEPARTURE_DATE_TIME between Departure_from and Departure_to 
    and upper(AD.NAME) like upper('%' || 
    Departure_airport || '%') 
    and F.CLASS = Class_id 
    and upper(AA.NAME) like upper('%' || 
    Arrive_airport || '%') 
    and F.CLASS = C.ID 
    and F.DEPARTURE_AIRPORT = AD.ID 
    and F.ARRIVE_AIRPORT = AA.ID;
```
where class_type and airport_type are object types which allow nesting of XML tags, and Departure_from, Departure_to, Class_id, and Arrive_airport are search parameters.

Executed Oracle function (defined within a package) queries Oracle tables with the passed input parameters, finds matching flights and sends them back to Java servlet engine. XML data are streamed as character stream into java.io.Reader object:

```java
java.sql.Clob clobResult = statement.getClob(1); // statement is an instance of Java.sql.CallableStatement
java.io.Reader xmlResult = clobResult.getCharacterStream();
```

It is important that database connection is not closed until the XML/XSLT transformation process is finished (because the CLOB Java stream will be closed, too) [3]. Java servlet determines client browser type by "user-agent" request header and loads an appropriate XSLT file. XSLT file for generating WML looks as follows:

```xml
...<xsl:output method="wml"/>
<xsl:template match="/">
  <xsl:text disable-output-escaping="yes">
    <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1_1.xml">
  </xsl:text>
  <wml>
    <card id="searchResults" title="Flight Search Results" newcontext="true">
      <p>
        <xsl:apply-templates/>
      </p>
    </card></wml>
  </xsl:template>

  <xsl:template match="/search_results/flight">
    <table columns="2">
      <xsl:apply-templates
        select="departure_date_time|class|arrive_date_time|departure_airport|arrive_airport|price_adult"/>
    </table>
  </xsl:template>

  <xsl:template match="/search_results/flight/departure_airport">
    <tr>
      <td><b>Departure</b></td>
      <td><xsl:apply-templates select="name"/></td>
    </tr>
  </xsl:template>

  <xsl:template match="/search_results/flight/arrive_airport">
    <tr>
      <td><b>Arrive</b></td>
      <td><xsl:apply-templates select="name"/></td>
    </tr>
  </xsl:template>
</xsl:template>
```

Java servlet sets the appropriate response content type regarding client browser. For "Mozilla" compatible browsers (like MSIE, Netscape, etc.) it sets response content type to "text/html" and prepares HTML output. For "Mobile" browsers (integrated into mobile phones) it sets response content type to "text/vnd.wap.wml" and prepares WML output. HTML and WML output represents transformed XML data (search result data) with the appropriate XSLT file (one for HTML content and one for WML content).

```java
boolean mobileUser = false;
String xslFilePath = null;
Document document;
// Set an appropriate content type
String browser = request.getHeader("user-agent").toLowerCase();
if(browser.indexOf("mobile") != -1) {
  response.setContentType("text/vnd.wap.wml");
  mobileUser = true;
} else {
  response.setContentType("text/html");
}

// Take an appropriate xslt file
if (mobileUser) {
  xslFilePath = getServletContext().getRealPath("WEB-INF/xslt/flight_search_wml.xsl");
} else {
  xslFilePath = getServletContext().getRealPath("WEB-INF/xslt/flight_search_html.xsl");
}
```

Transformation is done using:

- `javax.xml.parsers.DocumentBuilder`,
- `javax.xml.transform.Transformer` and
- `javax.xml.transform.dom.DOMSource`

```java
// Generating output
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
InputSource xml = new InputSource(xmlResult);
document = builder.parse(xml);
```

```java
// Use a Transformer for output
File stylesheet = new File(xslFilePath);
TransformerFactory tFactory = TransformerFactory.newInstance();
StreamSource stylesheetSource = new
```

StreamSource(stylesheet);
Transformer transformer =
tFactory.newTransformer(stylesource);

DOMSource source = new
DOMSource(document);
// Java servlet output
PrintWriter out = response.getWriter();
StreamResult result = new StreamResult(out);
transformer.transform(source, result);

Figure 2: Search results in Nokia Mobile
Browser Simulator

Figure 3: Search results in MSIE browser

VI CONCLUSION
As the speed of business increases, the requirement
for data interchange without human intervention
becomes a high priority requirement. Integrating
XML with Enterprise Databases and Application
Servers and XML-enabling their applications gives
businesses the infrastructure to satisfy this
increasing demand for access to and exchange of
information.
Oracle 10g provides a platform for natively
handling XML data and documents to Internet
standards and provides the development resources
and interfaces to fully integrate this support into
business applications.

REFERENCES
[1] Mark V. Scardina, Ben Chang and Jinyu
Wang, Oracle Database 10g XML & SQL:
Design, Build & Manage XML Applications in
Java, C, C++ & PL/SQL, McGraw-
Hill/Osborne, California, USA, 2004.
19991116.html.
[3] Eric Armstrong, Jennifer Ball, Stephanie
Bodoff, Debbie Bode Carson, Ian Evans, Dale
Green, Kim Haase and Eric Jendrock, The
J2EE™ 1.4 Tutorial, For Sun Java System
Application Server Platform Edition 8.1
2005Q2 UR2, Sun Microsystems, Inc.,
California, U.S.A.

Abstract: XML is widely used in Web publishing
systems. One of the reasons is that data in XML
format can be easily transformed using XSLT to
any Web publishing format that clients request.
This paper presents a method for generating and
transforming XML, for different content-publishing
purposes using Oracle 10g, Java and XSLT.

CONTENT-PUBLISHING APPLICATION
DEVELOPMENT USING ORACLE 10g, JAVA
AND XSLT, Daniela Nikolic, Drazen Nikolic