STUDIA INFORMATICA
Nr 1(10) Systemy i technologie informacyjne 2008

Presentation of relational and multimedia data using DAD access descriptors in Oracle 10g

Andrzej Barczak, Dariusz Zacharczuk
Institute of Computer Science, University of Podlasie
ul. 3 maja 54, 08-110 Siedlce

Abstract. The form in which information stored in a database is presented to the user is as important as the data itself. Every year WWW websites are becoming more and more popular and easily accessible way of data visualization. This chapter is devoted to the problem of access to Oracle database from the search engine. Therefore, we shall discuss issues concerning the access descriptor in Oracle 10g and data presentation (both relational and multimedia) using i.a. HTF and HTP packages.

Keywords. Oracle 10g, InterMedia, HTF, HTP

1 Introduction

In order to clearly picture the issue of data presentation by means of http protocol we will show, step by step, how to create an e-gallery using the tools provided by Oracle. The general procedure is as follows:

- Install database system,
- Install WWW server,
- Install necessary packages and tools,
- Configure WWW server,
- Create structure of tables for data storage,
- Create and run an online gallery.

2 Installation and configuration

Oracle 10g will be a database system used for the purpose of our project. We will need the following components:

- Oracle Database 10g release 2 (10.2),
- Oracle interMedia - Oracle 10g uses interMedia module, which conforms to SQL/MM standard, for handling multimedia data. SQL/MM (MM stands for MultiMedia) is also known as SQL Multimedia and

Application Packages. It extends SQL99 standard and supports new specific types dedicated for multimedia storage.

Oracle installation is a standard procedure if we use Universal Installer. While running the installation we should create a general use database (if we choose this kind of database, an example schemata install automatically). In this version of Oracle system (10.2), Oracle interMedia installs automatically and it is not necessary to upload the module from the producer's website.

In previous Oracle versions Apache - http server installed as default. In version 10g and subsequent it has been replaced by Oracle http Server with PL/SQL gate (mod_plsql) – it is an Apache module which allows for dynamic generation of www webpages using PL/SQL packages and stored procedures. It is an ideal solution for fast and flexible creation of web applications.

After successful server installation it is also necessary to install interMedia Code Wizard for PL/SQL gate. It will help us generate an example code for the necessary procedures. It consists of two parts: the Code Wizard package itself (ORDPLSGWYCODEWIZARD) and the tool package (ORDPLSGWYUTIL). It contains example tools that let us generate PL/SQL procedures for uploading and processing media data types stored in the database (using interMedia types.) This package must be installed on the ORDSYS schema.

```
SQL> @ordplsci.sql
SQL> @ordplsui.sql
```

The next step is creating a database access descriptor (DAD). Each query from a web browser directed to PL/SQL gate will contain a descriptor, with configuration parameters assigned to it that will be used for connecting with the database. The user who creates a descriptor must have the following roles assigned:

- CONNECT
- RESOURCE
- XDBADMIN

and the right to execute the SYS.DBMS_EPG package. Rights to execute CTXSYS.CTX_DDL are also useful. Additionally, we need to modify listener.ora file. The following must be added to the DESCRIPTION_LIST:

```
(DESCRIPTION=
      ( ADDRESS = (PROTOCOL = tcp)(HOST = localhost)(PORT =
8080) )
      (Presentation=HTTP)(Session=RAW)
 )
```

Built-in PL/SQL gate is implemented by DBMS_EPG package, which permits the web browser to invoke PL/SQL constituent procedures using HTTP Listener. The above code tells the Listener to listen on port 8080 and intercept http requests. It must be remembered that it is necessary to restart Listener after the above mentioned modification. Then, using the ORDSYS account (having granted appropriate rights), we create Database Access Descriptor:

```
BEGIN
  --the instruction removes the descriptor when needed
  --DBMS_EPG.DROP_DAD('ORDCWADMIN');

  --creating the descriptor: we enter the name
  --and access path (alias)
  --it is the only obligatory instruction when we want to
  --create any DAD
  DBMS_EPG.CREATE_DAD('ORDCWADMIN', '/ORDCWADMIN/*');

  --authorization makes it possible to execute requests
  --with rights of a selected user
  DBMS_EPG.AUTHORIZE_DAD('ORDCWADMIN', 'ORDSYS');

  --we set the method of user authorization
  --during attributes setting the name of the descriptor
  --is always the first argument, then we put the name
  --of the attribute and finally its value
  DBMS_EPG.SET_DAD_ATTRIBUTE('ORDCWADMIN', 'authentication-mode',
'Basic');

  --we set the main page (it will be the main page
  --of the installed Code Wizard package
  DBMS_EPG.SET_DAD_ATTRIBUTE('ORDCWADMIN', 'default-page',
'ORDCWPKG.MENU');
END;
```

When we create a DAD we enter the name of the object that will be used for any possible modifications of the descriptor's parameters. The path does not refer to the existing catalogues on the hard disk but it is a virtual alias that will be used to call individual packages or PL/SQL procedures. The www address for our descriptor looks as follows:

http://localhost:8080/ordcwadmin/

After entering it in the web browser window, we are requested for authorization (we log on as ORDSYS) and then we are re-directed to the previously set default page. If the default page has not been specified or we want another page to appear, it is necessary to add the name of the procedure, preceded, if necessary, by the name of the schema and the package, in which it is located, e.g.

http://localhost:8080/ordcwadmin/SCOTT.package.procedure

After logging on, we have a few options in our menu , from which we have to select 'Time zone management'. Setting Time zone is extremely important as regards proper interception of http headings by the procedures processing media. This operation may also be performed using the command line:

```
SQL> EXEC ORDPLSGWYUTIL.SET_TIMEZONE( server_gmtdiff=>-2 );
```

In this way, we have fulfilled four initial steps mentioned in the introduction. We have a database system together with the essential tools installed and we have also completed the process of configuration of both the packages and www server.

3 Creating a gallery

It is obvious that in order to display any data, it must be entered first. The initial step to create our gallery is testing the procedures responsible for loading media. All the files that PL/SQL gate receives are stored in so called document tables. Therefore, it is necessary to specify the tables for the documents (selected by the user) uploaded by the system (in this case – graphics files). It may be achieved using the DAD we created earlier or creating a new one. If we want to create a new descriptor, in order to be able to work with Code Wizard, we need to authorize it selecting the DAD authorization function from the main menu, and then we enter the alias (Important! We don't enter the name of the descriptor, only the access path) and the user, we approve the changes and choose option Change DAD. We log off and log on again.

We perform the operation of assigning the table containing the files received from HTML with DAD in the following way:

```
DBMS_EPG.SET_DAD_ATTRIBUTE('ORDCWADMIN', 'document-table-name',
'media_upload_table');
```

If we selected the table that already exists in the database, Code Wizard expects a table in the format:

```
CREATE TABLE document-table-name
( name           VARCHAR2(256) UNIQUE NOT NULL,
    mime_type      VARCHAR2(128),
    doc_size       NUMBER,
    dad_charset    VARCHAR2(128),
    last_updated   DATE,
    content_type   VARCHAR2(128),
    blob_content   BLOB );
```

We do not have to specify the existing table, the object will automatically be created by the Code Wizard during work.

However, we must create the target table, which will be used by our gallery while displaying data e.g.

```
CREATE TABLE galeria (
  id NUMBER PRIMARY KEY,
  opis VARCHAR2(30) NOT NULL,
  foto_min ORDSYS.ORDIMAGE,
  foto ORDSYS.ORDIMAGE );
```

Having created it, we return to the www panel. This time we will create a standalone procedure for loading data (create media upload procedure -> Standalone procedure). We select table GALERIA and go further. The configurator

asks about the document table mentioned above. Provided that the commands were run properly, the name GALERIA_UPLOAD should be displayed as default next to the option create table, which we should select and go further. In step three, we have to define the number of columns that will be uploaded simultaneously using HTML form. As we make an assumption that there are two types of images in our gallery: original and miniature ones, and we will create the miniatures ourselves (this will be discussed later) we only select the FOTO field. The next operation is selecting a column responsible for localization of a given picture in the table – we simply indicate the main key. The last option is selection of access to the table and data storage:

- Insert New row: always inserts a new row to the table
- Update existing row: always updates data in the table
- Conditional insert or update: if the row exists it is updated, if not a new one will be created.

We selected the first option. The penultimate step is selection of additional columns, which are to be stored together with media data and the selection of the procedure name for loading data. Here, the default options should be left unchanged. InterMedia also provides us with a possibility to create a procedure or only generate the source code (we choose code generation). We go further, check if all the options are entered correctly and click FINISH.

In the final stage we have an opportunity to have a look at the code (VIEW), so we click and save it in a separate file as it will have to be modified a little.

First, we remove In_ID from the header as we will create a sequence for generating main keys for our gallery:

```
create sequence seq_log_id start with 1;
```

We also modify the way of insertion and uploading data for updates from the target table. The miniature field should be taken into account and it should be initialized with the initial value in the following way:

```
INSERT INTO GALERIA ( ID, FOTO, FOTO_min, OPIS )
    VALUES ( seq_galeria_id.nextval, local_FOTO, local_FOTOm,
in_OPIS)
    RETURN id INTO in_id;

SELECT FOTO, foto_min INTO local_FOTO, local_FOTOm
   FROM GALERIA WHERE ID = in_id
   FOR UPDATE;
```

The full size image will be copied from the temporary table, however, we have to generate the miniature ourselves. In order to do this, we perform an operation on the variable storing our original image – we scale it and we call the function that automatically sets the attributes for newly generated data:

```
processCopy local_FOTO.processCopy( 'maxScale=75, 75',
local_FOTOm );
```

```
local_FOTOm.setProperties();
```

ProcessCopy function copies the image stored inside or outside the database into another picture stored in the database and executes the operations described by a given parameter (in this case – scaling) on the copy.

We may leave the remaining part of the code mostly unchanged.

The final stage of our work will be generating an interface for displaying the content of the gallery and a form for uploading pictures.

4 Creating WWW interface

Two packages: HTP (hypertext procedures) and HTF (hypertext functions) are used to generate HTML pages in Oracle. Both of them have identical methods names, identical attributes that are connected with specific methods, the only difference is that HTP procedures generate HTML code and the result is sent to the web browser, and functions that belong to HTF packed generate HTML code and return it in the form of a string of characters to the PL/SQL block. The description of all the available methods together with the examples may be found at: download.oracle.com/docs/cd/B19306_01/appdev.102/b14258/w_htp.htm#ARPLS391.

Let us create a form for loading images into our database using the options the package offers. Three things must be remembered:

1. POST must be the method of sending data in the form. GET method can accept up to 128 characters only.
2. target procedure (action) must be a procedure that was created earlier and modified according to our needs.
3. as we will send files, we must assign the value ‘multipart/form-data’ to the argument of the form. It means that the form will contain binary data – files.

Table 1. The procedure generating html form for sending files

PL/SQL code	HTML code generated by the procedure DODAJ.

PROCEDURE dodaj IS
BEGIN
  htp.htmlOpen;
  htp.headOpen;
  htp.title('Moja GALERIA');
  htp.headClose;
  htp.bodyOpen;
 	
  htp.formOpen(
  curl=> 'upload_procedure',
  cmethod=>'post',
  cenctype=>
  'multipart/form-data');
  htp.tableOpen(
  cborder=> 'border=1');
  htp.tableRowOpen;
  htp.tableData('Opis:');
  htp.tableData(
 	
  htf.formText('in_opis'));
  htp.tableRowClose;
  htp.tableRowOpen;
  htp.tableData('Foto:');
  htp.tableData(
  htf.formFile(
  cname=>'in_foto'));
  htp.tableRowClose;
  htp.tableRowOpen;
  htp.tableData('');
  htp.tableData(
  htf.formSubmit(
  '-ZAPISZ-'));
  htp.tableRowClose;
  htp.tableCLose;
  htp.FormClose;
 	
  htp.BodyClose;
  htp.HtmlClose;
END dodaj;
	<HTML>
	<HEAD>
	<TITLE>Moja GALERIA</TITLE>
	</HEAD>
	<BODY>
	
	<FORM
	  ACTION="upload_procedure"
	  METHOD="post"
	  ENCTYPE="multipart/form-data">
	
	<TABLE border=1>
	
	<TR>
	<TD>Opis:</TD>
	<TD>
	  <INPUT TYPE="text" NAME="in_opis"></TD>
	</TR>
	<TR>
	<TD>Foto:</TD>
	<TD>
	  <INPUT TYPE="file" NAME="in_foto"></TD>
	
	</TR>
	<TR>
	<TD></TD>
	<TD>
	  <INPUT TYPE="submit"
	  VALUE="-ZAPISZ-"></TD>
	</TR>
	</TABLE>
	</FORM>
	
	</BODY>
	</HTML>

Summing up, the procedure generating the form for sending files to the server should look more or less the same as it is shown in Table 1. The fields of the form in the example are additionally placed in the table in order to make the whole code clearer and more transparent.

The last stage is displaying the image. As we have both the original image and its miniature in the database the gallery will display small-sized images, which will be references to actual files. HTML link is generated by the procedure htp.anchor:

```
htp.anchor('POKAZ_ZDJECIE?id=' || v_id,
         wyswietl_miniaturke(id, height, width));
```

The first parameter is WWW address. POKAZ_ZDJECIE is a procedure which will display the original image. It has the main key as the parameter that indicates which image should be displayed. The second parameter is the object we should click – in our case it will be the image miniature. The function wyświetl_miniaturke uploads the essential data to display graphics: identifier, height and width (the image size can be read out using the attributes height and width of the ORDImage object e.g. foto.height).

The function that displays the miniature must contain (in its body) an instruction of the following form:

```
RETURN htf.img( curl=>'DAJ_MEDIA?id=' || v_id,
                cattributes=>'height= '|| v_h || ' width=' ||
               v_w);
```

which will return HTML code necessary for miniature display. Simultaneously, the DAJ_MEDIA method supplies binary data, which is accepted as the first attribute of the function generating HTML code for displaying the image on the WWW webpage. The body of DAJ_MEDIA must then contain:

- uploading binary data about an image with a given identifier from GALERIA table:

```
SELECT foto_min INTO local_image FROM galeria WHERE id = v_id;
```

- checking the date of the file modification, if the data is unchanged the web browser will use the files present in cache memory. The checking functions (which is easy to notice) come from the Code Wizard package.

```
IF ordplsgwyutil.cache_is_valid( local_image.getUpdateTime() ) THEN
     owa_util.status_line(ordplsgwyutil.http_status_not_modified);
     RETURN;
END IF;
```

- it the data has been modified, we send the heading MIME and fetch the image to the web browser. MIME type is stored as one of the attributes of ORDImage type. Here, we must also use another package that has not been discussed yet, namely, OWA_UTIL. It contains tools for the management of environment variables CGI, displaying data returned to client and displaying the results of queries in a HTML table.

```
-- generates content-type field of HTTP header
owa_util.mime_header( local_image.mimeType, FALSE );
--sets the date of the last modification of cache memory
--to the date that is consistent with the file
ordplsgwyutil.set_last_modified( local_image.getUpdateTime() );
--closes HTTP header
owa_util.http_header_close();
```

- and the last requirement for the DAJ_MEDIA procedure – transfer of binary data. Package, which provides the interface for uploading files, BLOBs and data of BFILE to WPG_DOCLOAD type. We will gain access to binary data (BLOB) calling localData attribute of the ORDImage structure.

```
wpg_docload.download_file( local_image.source.localData );
```

There is one more procedure to discuss that we have used, namely: POKAZ_ZDJECIE, which, may well use the procedure wyswietl_miniature. It just has to be parameterized in order to distinguish original images from reduced size images. This, however, exclusively requires quite simple programming skills, which everyone should be able to deal with.

5 Summary

Oracle is undoubtedly a powerful system. It has countless tools, which not only help in effective management and administration of a database but also – as we have just experienced – makes it possible to present this data in an easy way.

Creating www websites using HTP, HTF or OWA_UTIL packages is only a little more complicated than using such technologies as PHP for the same purpose. We may also risk the assumption that the PL/SQL code that generates HTML code is clearer and more readable than the generated code.

If we configure Oracle properly, the presentation of multimedia data or text data is not a difficult thing, and we hope that after reading the paper it will not present problems to anyone.

References

1. Krzysztof Jankiewicz, Marek Wojciechowski: Standard SQL/MM: SQL Multimedia and Application Packages, http://WWW.cs.put.poznan.pl/mwojciechowski/papers/sem04.pdf
2. Oracle Database PL/SQL PAckages and Types Reference, 37 DBMS_EPG http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14258/d_epg.htm
3. Oracle FAQ'a, http://WWW.orafaq.com/faqmodpl.htm
4. Oracle Multimedia Software Downloads, http://WWW.oracle.com/technology/software/products/intermedia/index.html