"1st International Symposium on Computing in Informatics and Mathematics (ISCIM 2011)"
in Collabaration between EPOKA University and "Aleksandër Moisiu" University of Durrës
on June 2-4 2011, Tirana-Durres, ALBANIA

Multimedia Database Systems and Oracle Video Streaming

Silvana GRECA[1], Denada XHAJA[2], Eriglen GANI[3]

[1,2,3] *Department of Informatics, Faculty of Natural Sciences, University of Tirana, Tirana-ALBANIA*
Email*: silvana.greca@depinfo.info, dxhaja@gmail.com, neglire@gmail.com*

ABSTRACT

The acquisition, generation, storage and processing of multimedia data in computers and transmission over networks have grown tremendously in the recent past. A multimedia database management system must support multimedia data types in addition to providing facilities for traditional DBMS function like database creation, data modeling, data retrieval, data access and organization, and data independence.

In this paper we present several aspects of multimedia databases. Most of the classical database systems are extended to support the storage of multimedia data. This paper will focus on oracle database and in its datatypes for supporting the storage and management of multimedia data, known as Intermedia. It is also shown the architecture, configuration and implementation of video streaming. All the multimedia data are stored inside the oracle database so they will be under transactional control. The implementation uses several tables to demonstrate all the process for doing it in the Oracle database with Helix server. We have created the procedures in PL/SQL for inserting, retrieving the video and we have also used a PL/SQL package integrated with PHP script in order to help users to make searching using keywords.

Keywords: multimedia database, data type, ordvideo, helix server, streaming, procedure, oracle

1. INTRODUCTION

A Multimedia Database Management System must support multimedia data types in addition to providing facilities for traditional DBMS functions like database creation, data modeling, data retrieval, data access and organization, and data independence [1]. The contents of MMDBMS are: media data, media format data, media keyword data and media feature data [2]. The huge amount of data in different multimedia-related applications warranted to have databases as databases provide consistency, concurrency, integrity, security and availability of data, query and retrieval of data.

Oracle interMedia is a feature that enables Oracle Database to store, manage, and retrieve images, audio, video, or other heterogeneous media data in an integrated fashion with other enterprise information [3]. It extends Oracle Database reliability, availability, and data management to multimedia content with the new data types, ORDImage, ORDAudio and ORDVideo [3].

In this paper we show the implementation of an application which uses oracle database and helix server to do video streaming. The application provides the management (insert, update, delete) of different subject, their topics and the videos for each topic, considering this application as a teleteaching environment.

2. THE CONFIGURATION OF ENVIRONMENT AND THE DATABASE

In this project we have used oracle server 10g r2 on windows xp sp2 and oracle database client v11.1.0.6. The directory Bin of oracle and the path of client are included in the Path to Environment Variables of operating systems. Helix server is a program which is able to do streaming video and audio. It is the most powerful server software for streaming media [4]. Oracle Multimedia Plug-in 3.0 for RealNetworks Streaming Server enables the connection between Oracle Server and Helix Server so by making possible video streaming directly from the Oracle database to a client. The most important point is the configuration of a wallet, which allows the connection to Oracle Server without the need of writing username and password. In this way is achieved a higher security. After creation of the wallet it is necessary to modify the sqlnet.ora file in a way that any attempt for login must pass from the created wallet. In the same way we must configure the tnsnames.ora file where we add some other lines for the alias created at the wallet. Then we have configured a mount point in Helix Server which will be the point to communicate with oracle server. We have created a role Menaxher, given to it the necessary privileges and granted to it the users. For the application we have also used Wamp server, Zend Studio, Php, realplayer and totalvideo converter.

The most important tables of our database are: "Lendet" (subject), "Leksionet" (topics) and "Videot" (videos). One subject can have more then one topics and a topic may have more then one video related to it. We have also created the appropriate tables for roles and users. The most special table is "videot" because is used the ordvideo datatype:

```
create table videot
(id number not null primary key,
leksion_id number not null,
video ordsys.ordvideo not null,
data date default sysdate not null
constraint fk_videot_leksionet
foreign key(leksion_id) references leksionet(id) on delete
cascade);
```

A video is stored in a field of type ORDSYS.ORDVideo. This is not a primitive type, but an aggregate type created by this syntax: CREATE TYPE ORDVideo AS Object ... [5]. This type has some attributes and methods. It is necessary to store the binary information of the video and some other information such as: number of frames, the video duration, the frame rate, the dimension of the video, the number of colours, the resolution of the frame, the format of videos, MIME type. Video is stored in ORDSource which has some attributes where the most important is the field that actually saves the video of type BLOB.

3. GENERATION OF VIDEO STREAMING

3.1 Video streaming architecture

There are two ways to view a media file which is downloaded from the internet. The first way concerns to downloading all the media file from the internet and then the file can be seen, and the second way has to do with the viewing while it is downloading into the buffer memory of the computer, therefore it can be seen almost simultaneously with the downloading from the internet. There are also two ways to store the information of media type, such as audio and the video, which are: either as file or in a database. In this paper the project deals with the video streaming with information obtained from the database. The request for streaming is made by the client, when he writes an address (URL) in any program that reads audio or video media files such as Windows Media Player, QuickTime, RealPlayer, or from any Web Browser. The request for video streaming is sent to the server on the port 8090 (arbitrarily defined) with the HTTP protocol (the case of Web Browser), or in the port 554 through RTSP protocol. Helix Server that listens to these two ports, after receiving the request, calls the Oracle plug-in that is as an intermediary between Helix Server and Oracle, and sends a request to get the video. The Helix understands that this request for video is directed to Oracle, from the name of the mount point specified in the URL. From Helix, the Oracle plug-in is given the ID of the video, the name of the PL/SQL procedure that will make query to the database, and the Database Connection String as is shown in figure 1. The procedure makes a query to the relevant table to get the video.

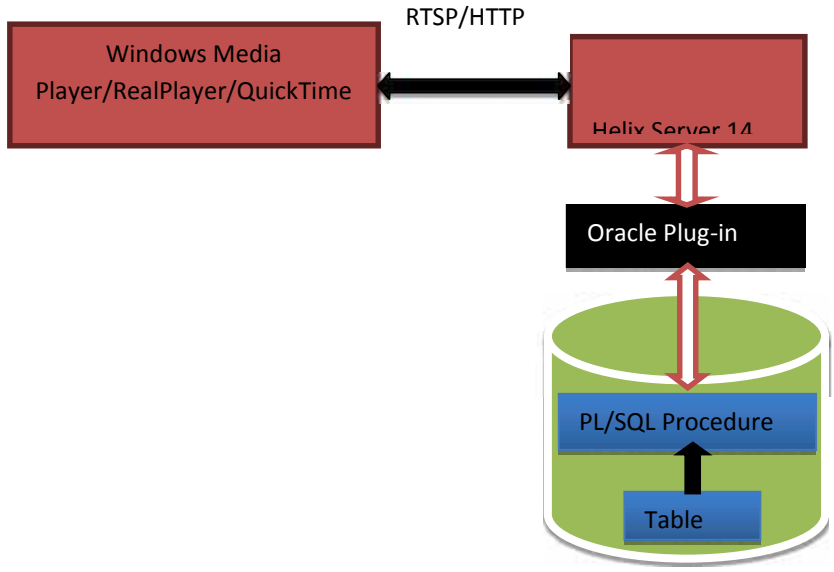

Figure 1 The architecture of video streaming using helix server and oracle server

The procedure which take the video is:

```
Create or Replace Procedure MerrVideo(idin in integer, TeDhena out blob,
TipMime out varchar2) as
        obj ordys.ordvideo;
        tmp blob;
        mime varchar(200);
        format varchar2(31);
        ctx raw(64) :=null;
Begin
        Select video into obj from videot
        where id  = idin;
        dbms_lob.createtemporary(tmp, true, 10);
        obj.getContentInLob(ctx, tmp, mime, format);
        TeDhena:= tmp;
        TipMime:= mime;
        COMMIT;
Exception
          when no_data_found then dbms_output.put_line('Video me këtë ID nuk
ekziston');
end MerrVideo;
```

This procedure has three parameters, an IN which gets the id of the video, and two OUT. Initially, it makes a query to the table that holds the videos, and the result, if any, is placed on the variable "obj", if there is no result then the exception is called implicitly. Next, we create the "tmp" variable with

CreateTemporary procedure of the package DBMS_LOB, which means that the value that will hold this variable will exist as long as will exist the session, so the value of the last parameter is 10, otherwise the value of this variable will exist until the end of this function call. Afterward, we use the method getContentInLob which takes 4 parameters. From this method of ORDVideo, we get the video from the Blob of ORDSource and the Mime type. These values are assigned to the last two parameters of our procedure. When the request is made by the browser, Helix Server generates a metadata information, a file with the extension Ram. This file shows to the browser the MIME type of multimedia information, which was taken from "MerrVideo" procedure. Based on the MIME type, the browser calls the appropriate program's plug-in which is able to interpret the information. In this project, we have used video with RealVideo format. When the browser obtains the Ram file, it calls the RealPlayer plug-in, and then begins to display the video (in a section within the website). If the RealPlayer is not installed, the video is not displayed. We can write:

rtsp://localhost:554/live/23
http://localhost:8090/ramgen/live/23

When it is made a request for video, in the URL that uses HTTP protocol, the word ramgen commands the Helix to generate the metadata Ram file.

3.2 Architecture of the application and implementation

This application, which is a web based, displays in a web browser a menu with the list of subjects and the list of topics for each subject (after clicking one of the subjects). When it is clicked one of the topics, are displayed some "links" with the videos of the appropriate topic. A topic may contain more than one video. The application also provides the opportunity to add topics or other subjects, all these may be generated dynamically.

Figure 2 shows that two different server applications send data to the client browser, Helix Server which does the video streaming, so inside the browser it downloads and displays the video, while the other, Apache Server generates dynamically the data received from Oracle, the php files that are on the server generate dynamic data.

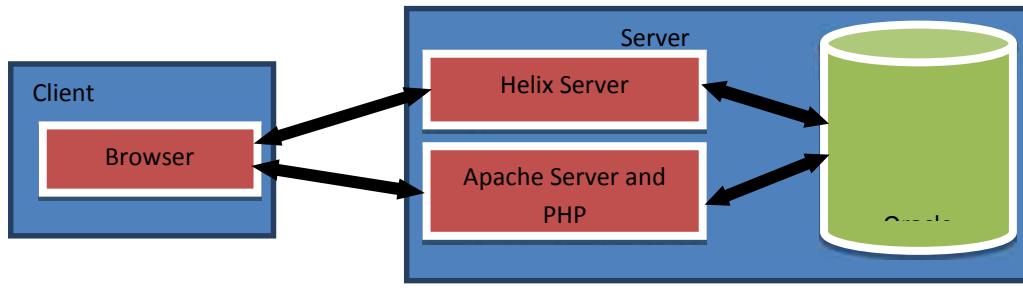

Figure 2 Architecture of application

These Php files read the database tables, and display in the client browser the links for the menu of the list of subjects, topics and links of the appropriate videos, while Apache Server is doing the interpretation of these Php files. This way facilitates the users, just with a few clicks from their browser, they display the respective video, without the need to write the URL of the video, while Apache Server and Php generate the appropriate link of the video. It is important to mention that both server applications, Apache Server and Helix Server are located in the server and communicate with the Oracle database.

Initially we display a menu with the list of subjects, in a way that will be more easily for the user to choose the right topic and then the related videos. This is done by the page "menu.php".

A link from the dynamic generation is:

Baza të dhënash

When this link is clicked, in the HTML frame called "trupi", all the topics of the subject "Baza të dhënash" are displayed. When the user clicks on one of the topics, it is called another web page named "leksione.php" which displays all the related videos.

The most important fragment of the code of the page "lenda.php" is:

```
<?php
$lidhja = oci_connect(videostream, videostream, 'orcl');
$query = oci_parse($lidhja, "Select id, emri_leksionit From leksionet Where
lenda_id = :t_id;");
$id = $_REQUEST['id'];
oci_bind_by_name($query, ':t_id', $id);
oci_execute($query);
while($fusha = oci_fetch_array($query)) {
echo "<a href='leksioni.php?id=".$fusha['id']."' target='_self'> " .
$fusha['emri_leksionit'] . "</a><br /><br />";
}
oci_close($lidhja);
?>
```

The query has a condition, which should be given a dynamic value, to get only the topics of the clicked subject. This value is get from the Php variable that holds the *id* of the subject clicked by the user. The clicked link generates an URL such as http://localhost/lenda.php?id=5. Through $_REQUEST we get the value of the *id* parameter of URL, and assign it to the variable $id. The query is executed, and then through the while cycle are generated the topics. The user can click on each of the generated links and then it is directed to another page called "leksioni.php", which displays all the videos of the topic. The most important part of the code of page leksioni.php is given:

```
<?php
$lidhja = oci_connect(videostream, videostream, 'orcl');
$query = oci_parse($lidhja, "SELECT id FROM videot WHERE leksion_id
= :t_id;");
$id = $_REQUEST['id'];
```

```
oci_bind_by_name($query, ':t_id', $id);
oci_execute($query);
while($fusha = oci_fetch_array($query)) {
?>
<embed src=http://localhost:8090/ramgen/live/<?php echo $fusha[‘id’]; ?>
type="audio/x-pn-realaudio-plugin" autostart=”false” hidden=”false”
loop=”false” volume=”50”></embed>
<?php
}
oci_close($lidhja);
?>
```

The *embed* tag serves to display the video in the relevant application which is called by the browser based on mime type define by the type attribute. The link of the video is assigned to the *src* attribute where it is specified and the id taken from the query. When the browser makes a request with this URL, Helix server takes the *id* located at the end of the URL. The helix gives this *id* and the name of the procedure to Oracle plug-in, which should be called to stream this video. Figure 3 display the video streaming for a topic and figure 4 display topics details for a specific subject.

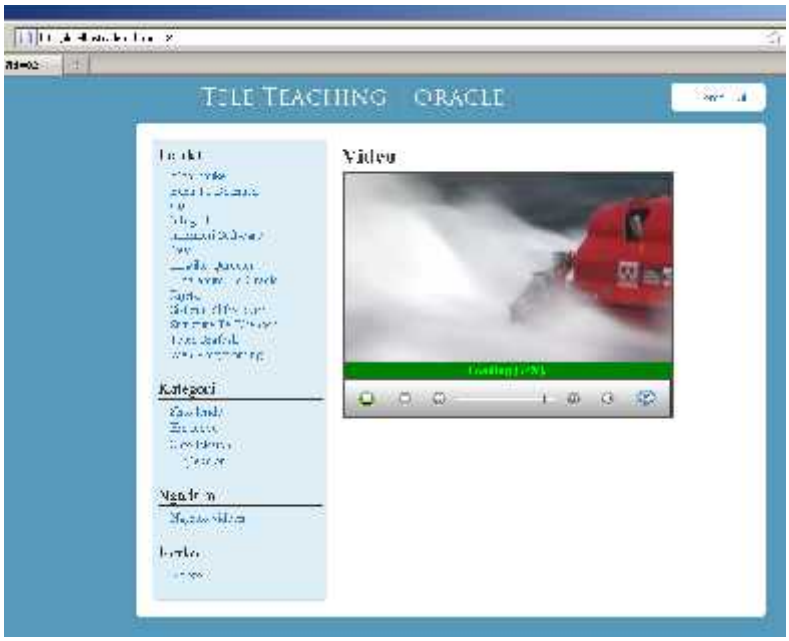

Figure 3 Video streaming

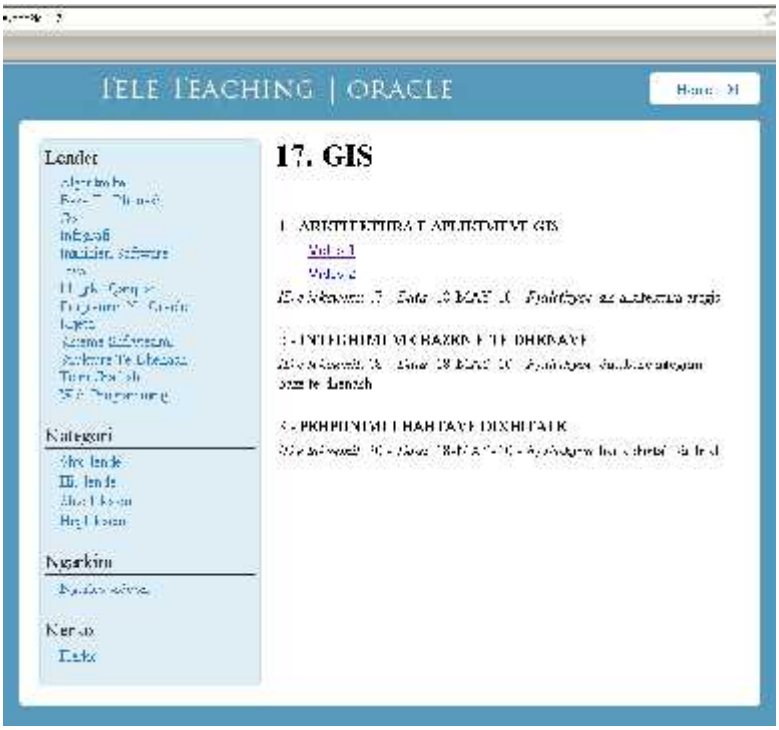

Figure 4 The details of topics for a subject

3.3 Searching the topics by the title and keywords

To search for a topic (indirectly this is a search for the video) we have used two elements: the title of the topics and keywords. The keywords must be separated from each other only by space, and should not be used characters like **ë** or **ç**. To accomplish this we have used two procedures, one is used to search for words in the title, while the other deals with the search of words placed in the keywords field. These procedures display in browser through Php a list of topics that contain words that the user is searching, either in the title or in the keywords. A significant part of this work is done from the Php script in the page "kerko.php". The results will be displayed, if something is found. The written words will be search initially as a hole, than if there are more than one word, each of them will be placed into a Php array, and will be compared to the topics and the keywords. This algorithm is oriented by the Albanian language. In this kind of search, names are used more often than verbs. In Albanian language, the names change the suffix in a different way from those in English. For this reason, in general the names do not change by more than two letters assuming that the root of word in Albanian do not change, there are also exceptions from the general case. According to this algorithm, words with one or two characters will not be taken in consideration, because they are not considered as "words" and the searching is done by deleting one, two or three characters from the word.

4. CONCLUSION

In this paper we have stored the video data in database, this has some advantages: the data are kept synchronized, secure and as is shown in this implementation, the media data can be searched easily. We have shown how to implement video streaming based on Oracle10g and Helix Server. For doing this, we use Oracle interMedia Plug-in 3.0 for the connection between Oracle database and Helix Server as is shown in figure 1. Next, it is shown the architecture of application in figure 2, and the implementation of video streaming. The implementation is done by using pl/sql procedures and php scripts. The user has the possibility to search the database by the topics or keywords to find the appropriate topics. The system and application improves the management, security and convenience of video data by using the object type ORDVideo. Nowadays, the size of database is not more a problem because of advances in hardware technology.

REFERENCES

[1] H. Kosch and M. Döller, "Multimedia database systems: Where are we now?," Special Session Talk to be given at the IASTED DBA-Conference in Innsbruck, February 2005.

[2] Keerthana, Kiran (2010). Multimedia Databases. http://blog.unsri.ac.id/userfiles/

[3] Oracle® interMedia User's Guide 10g Release 2 (10.2) B14302-01, June 2005

[4] Helix systems integration guide. ©2010 RealNetworks, Inc.

[5] Oracle® interMedia Reference 10g Release 2 (10.2) B14297-01, June 2005