Functionality of a Database Kernel for Image Retrieval

Cosmin Stoica Spahiu, Cristian Marian Mihaescu, Liana Stanescu, Dan Burdescu, Marius Brezovan
University of Craiova
Faculty of Automation, Computers and Electronics
Craiova, Romania
{stoica.cosmin, mihaescu, stanescu, burdescu, marius.brezovan}@software.ucv.ro

Abstract—**This article presents a software tool that implements a dedicated multimedia database management server for managing alphanumerical and multimedia data collections from medical domain, along with the file structure used. The server is designed to manage medium sized personal digital collections. The recognized commands and the functionality of the server are also presented. An element of originality is that along with classical operations for databases, it includes a series of methods used for extracting visual information represented by texture and color characteristics. The extracted data are stored in the database in a special data type called IMAGE, with a specific structure that can be used for visual queries. Some clustering algorithms are used to increase the image retrieval speed.**

Keywords - multimedia; database management server; content-based retrieval; clustering; file system.

I. INTRODUCTION

Nowadays information is not limited only to strings. In order to facilitate different multimedia processing, advanced database management systems can integrate various data types (such as: images, video, text and non-numeric information) in a single database.

The aim of a database is to offer the user the possibility to use it for the query process. There are two types of queries that can be executed. One type of query process is the classical one that use simple text-based query. It can be used in the following cases:

a) The doctor knows the name of the patient and he wants to find all information about him

b) The doctor knows a certain diagnosis and wants to find all similar cases with the same diagnosis.

Another modern type of query is the content-based retrieval. That means that the search is made using similar characteristics of an image. The images have to be processed and extracted the characteristics. These characteristics will be compared later in order to find the images most similar to the query image. As an immediate effect the traditional information management systems cannot be used on this large collection of various data types [1][2].

The information in raw format is usually useless. The real benefits come when different decisions can be taken based on it, or it can be explored and obtained provenience information.

There are some domain areas were large volumes of information are stored in centralized or distributed databases. Few of these domains, are: digital libraries, image archives, bioinformatics, imagistic medicine, health, finances and investments, production, marketing strategies, telecommunication networks, scientific areas, WWW and biometry.

Usually in these databases the multimedia information (images, video, audio, etc.) is stored either in separate files, or inside the database in BLOB data types. This solution is not suitable all the time because makes it difficult to execute directly the visual content-based retrieval queries.

All these problems presented above are leading to the necessity for developing a multimedia database server that include intelligent data storage and processing methods used for content-based retrieval and automate data classification.

The solution proposed in this paper also uses a clustering method for grouping similar images.

Clustering is the process of grouping a set of physical or abstract objects into classes of similar objects. As a product of clustering process, associations between different actions on the platform can easily be inferred from the logged data. In general, the activities that are present in the same profile tend to be found together in the same session [3].

This paper presents a dedicated database management server (DBMS) that is based on the relational model. The DBMS can be used for managing medium sized databases from medical domain. For clustering process it is used Weka package.

The paper is structured as follows: in Section 2 there are presented similar implementations of well known servers, along their particularities. Section 3 presents the architecture of the server. Section 4 presents the database management and querying operations, and in the Section 5 it is presented a solution for clustering the images.

II. RELATED WORK

Most of the applications that use multimedia information are based on traditional database management systems as MS SQL Server, My SQL or Interbase. Each of them offers partial support for multimedia content.

MYSQL offers only the BLOB data type that can be used to store images. A BLOB is a binary large object that can hold a variable amount of data. BLOB attributes have no character set, and sorting and comparing operations are based on the numeric values of the bytes [1] [4].

MS SQL Server offers a data type called "image", but with no other support. It is considered a variable-length binary data having the size between 0 and 2^{31}-1

(2,147,483,647) bytes. There are no pre-defined functions that can be used for extracting the characteristics or building content-based queries. More than that, in the MS SQL Server 2008 it is specified that "ntext", "text" and "image" data types will be removed in a future version of Microsoft SQL Server. The recommendation is to avoid using these data types in new development work, and to plan to modify applications that currently use them. It should be used instead: nvarchar(max), varchar(max), and varbinary(max) [5] [6].

The complete solution is provided by Oracle - the Oracle 10g database server and Intermedia tool that can manage all kind of multimedia data, including DICOM files [7][8] [9].

In addition to the image support offered via the ORDImage object type, in Oracle 10g version interMedia provides support for the first edition of the ISO/IEC 13249-5:2001 SQL/MM Part 5: Still Image Standard. This standard defines object relational types for images and image characteristics. Each object type includes attributes, methods, and SQL functions and procedures. The use of the SQL standard interface may make some applications to be more portable across various vendor databases.

For the clustering algorithms, there are many methods detailed in the literature: partitioning methods, hierarchical methods, density-based methods such as [10], grid-based methods or model-based methods. Hierarchical clustering algorithms like the Single-Link method [11] or OPTICS [12] compute a representation of the possible hierarchical clustering structure of the database in the form of a dendrogram or a reachability plot from which clusters at various resolutions can be extracted.

Because we are dealing with numeric attributes, only the iterative-based clustering is taken into consideration from the existing partitioning methods. The classical k-means algorithm is a very simple method for creating clusters. Firstly, it is specified how many clusters are being thought: this is the parameter k. Then k points are chosen at random as cluster centers. Instances are assigned to their closest cluster center according to the ordinary Euclidean function. Next the centroid, or the mean, of all instances in each cluster is calculated – this is the "means" part. These centroids are taken to be the new center values for their respective clusters. Finally, the whole process is repeated with the new cluster centers. Iteration continues until the same points are assigned to each cluster in consecutive rounds, at each point the cluster centers have stabilized and will remain the same thereafter [10][11][12].

From a different perspective there may be computed the following parameters for a cluster: means, standard deviation and probability (μ, σ and p). The expectation-maximization (EM) algorithm that is employed is a k-means clustering algorithm type. It takes into consideration that we don't know either parameters. It starts with initial guess for the parameters, it uses them to calculate the cluster probabilities for each instance, it uses these probabilities to estimate the parameters, and repeat. This is called the EM algorithm for "expectation - maximization". The "expectation" is the first step needed to calculate the cluster probabilities (which are the "expected" class values); the "maximization" of the likelihood of the distributions given is the second step which it is needed to calculate the distribution parameters [12].

The quality of clustering process is measured by computing the likelihood of a test data set given from the obtained model. The goodness-of-fit is measured by computing the logarithm of likelihood, or log-likelihood: the larger this quantity is, the better the model fits the data. Instead of using a single test set it is also possible to compute a cross validation estimate of the log-likelihood.

III. DATABASE KERNEL – OVERVIEW

The kernel it is a tool that can be used for database creation, maintenance, and executing simple text-based queries or content-based visual queries. The multimedia digital collections from medical domain are used for these queries.

This dedicated MMDBMS permits database creation, table and constraints definition (primary key, foreign keys), inserting images and alphanumerical information, simple text-based queries and content-based queries using color and texture characteristics. The software tool is easy to be used because it respects the SQL standard and has the advantage of low cost. The users do not need advanced informatics knowledge. It is a good alternative for a classical database management system (MS Access, MS SQL Server, Oracle10g Server and Intermedia), which would need higher costs for database server and for designing applications for content-based retrieval [13].

The Figure 1 presents the general architecture of the medical system.

In the first step the client application that uses the server should connect to the database. This way it will be created a communication channel between them. All commands and responses will use this channel to send queries requests and receive answers.

The server has two main modules: kernel engine and database files manager.

The kernel engine includes all functions implemented in the DBMS. It is composed from several sub-modules each of them with specific tasks:

- The main module. It is the module which manages all communications with the client. It is the one that receives all queries requests, check what the type of requested query is, extracts the parameters of the query and calls the specific module to execute it.

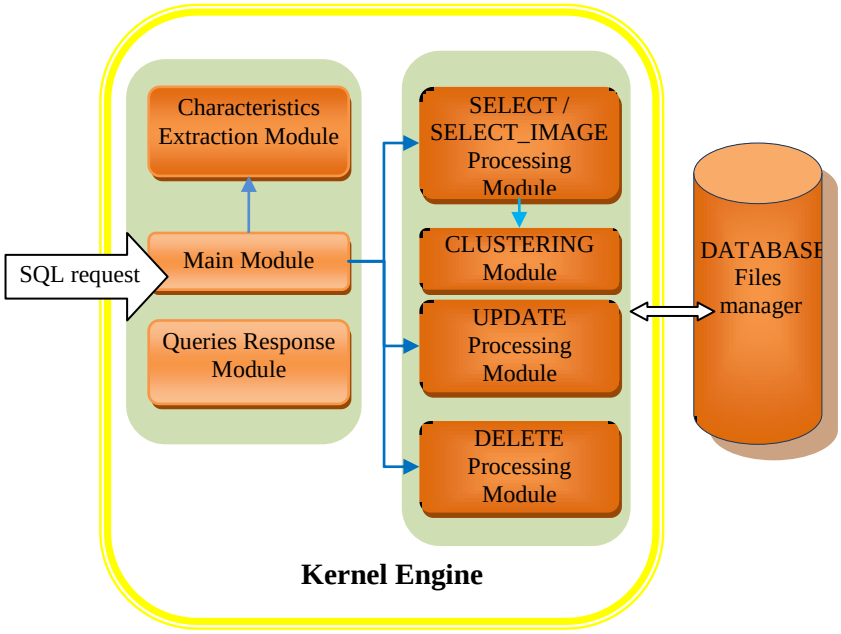

Figure 1. General architecture of the system.

- Queries response module. After the query is executed, the results will be sent to the Queries Response Module. It will compact the result using a standard format and then return it to the client. The client will receive it on the same communication channel used to send the request.
- Select/Select/Image Processing module. If the main module concludes that is a SELECT SQL command, it will call the Select Processing module. This module extracts the parameters from the query and then search in the database files for specific information. If the query is a SELECT IMAGE query, it will use for comparation the similitude of characteristics instead equality of parameters.
- Characteristics Extraction Module. When the main module receives a SELECT IMAGE or a UPDATE query which uses an image that is not already in the database it is needed first to process it. This module is called to extract the color and texture characteristics of the image. The data of the results will be used to initialize a variable of IMAGE data type.
- Clustering Module. The role of this module is to assign each image to a cluster containing similar images.
- Update Processing Module. When the query received from the user is an UPDATE command, it will be called this module to execute it.
- Delete Processing Module. It is called when the user executes a DELETE command. The kernel executes only logic deletes. It never executes physical deletes. The physical deletes are executed only when a "Compact Database" command is sent by the user.

The second main module is the Database Files Manager. It is the only module that has access to the files in the database for reads and writes. It is its job to search for information in the files, to read and write to files and to manage databases locks. There are two kinds of locks: shared locks (read locks) and exclusive locks (write locks). A read lock it is enabled when a client module requests a read from a file (that represents a table in the database). All other read requests will be permitted but no writes will be allowed. If the client module request a write access, it will be enabled a write lock. No other requests will be allowed until the lock is canceled.

The results will always be returned to the module which made the request. The data read or wrote to files is not structured in any way. This module does not modify the data structure in any way. All the results will be raw data, as it is read from the files or received from client modules.

IV. DATABASE MANAGEMENT AND QUERY

This system is a client-server tool that can be used by all kind of applications that work with databases. The client applications can be implemented in C++, Java or any other language that supports TCP/IP communication [1][2].

An element of originality for this tool is that among the classic operations that can be executed (like databases creation, maintenance, simple text-based query) it is included a new module for content-based visual query using color and texture characteristics. This module will search in the database for images with similar texture and color pattern and return all the associated information (the similar image along with the rest of the attributes).

The first step in the communication between client-server is to establish the TCP connection. Only after the client can send query requests. The connection is made using TCP

standard and should include a username and a password for security. Each user has defined a series or rights that he can benefits from: create databases, create tables in a specific database, modify tables' structures, insert data into tables and execute queries. Each of these rights is defined at the database level. When a database is created, only users who have defined rights can work with it.

The implemented commands are related to: listing the available databases, create/delete a database, create/delete a table, adding constraints for tables (primary key, foreign key), etc. Another element of originality for the server is that the images can be seen automatically, when browsing the records in the database.

The commands categories recognized by the system are:

A. Managing Databases

1. *create database <database_name>*

eg: create database student

This command creates a new database on the server.

2. *use database <database_name>*

e.g.: use database student

Before being able to execute operations on it, first it should be defined the default database.

3. *create table <table_name> (<atribute1> <data_type1>, <atribute2> <data_type2>, [<atribute3> <data_type3>,…])*

e.g.: create table person (id int, name varchar(20), age double)

create table image (id int, name varchar(20), picture image)

The command will create a new table in the database with the specified attributes. A table there can have any number of attributes. The data types that can be used are:

- integer
- double
- varchar
- image

If it is used the varchar data type, it is needed to be specified the maximum size of the array.

4. *alter table <table_name> add primary key (<attribute_name>)*

e.g.: alter table person add primary key (id)

The command will define <attribute_name> as primary key in <table_name>.

The composed primary keys can be defined by using this command for each attribute in the key.

5. *alter table <modified_table_name> add foreign key (<atribute1_name>) references <table_name2> (<refered_ atribute2_name>)*

e.g.: alter table person add foreign key (id) references picture(id)

The command defines a foreign key between two tables <modified_table_name> and <table_name2>. The key will link the two attributes <atribute1_name > and <refered_ atribute2_name>.

The many-to-many links cannot be represented, as the server respect the relational model. That is why at least one of the attributes which is part of the external key, must be a primary key.

6. *get table keys <table_name>*

e.g.: get table keys person

This command returns all the references defined on all of the attributes existing in the table

B. Inserting Data into the Database

The communication between client application and DBMS is based on messages exchange. All messages represent commands written in SQL language.

1. *insert into table_name values (value1, value2, [value3, …])*

There are two possible cases:

a) *Inserting only text:*

e.g.: insert into PATIENT values("disease", "polyps")

In this case the main module receives the command, and checks the user's rights. If the user has enough rights to insert new values, it calls the Update Processing Module to add data into the database.

b) Inserting text and images

e.g.: insert into person values (1,'George O.',20.5)

insert into image values(1, 'George O.','analysis.bmp')

The values between parentheses will be added in the table <table_name>. It is mandatory to add values for each attribute. There should be no NULL attributes. The order of the inserted values should be the same as the one of the attributes in the table.

For this example it is called first the Update Processing Module. It checks the metadata of the table and finds out that one of the attributes has the type "image". It notifies the Main Module that it should receive an image file from the client called "analysis.bmp". After the image is received, it will call the Characteristics Extraction Module to process the image and create the image type value. The record is inserted into database only after this type is created.

In order to increase the retrieval speed for future queries, it is also called the Clustering Module. It will assign the new image to the cluster that contains images with similar characteristics.

C. Simple Text-Based Queries

The text-based queries can be used on any attribute defined in the table. It is used the SQL syntax, having the possibility to specify to return only certain attributes from a record, or all the attributes. There are accepted any number of conditions specified in the WHERE clause, which are combined by AND or OR operators.

The syntax is:

*select * from table_name where attribute1 [</=/>] values [and attribute2 [</=/>] values2 …]*

e.g.: select * from person

select * from person where age>45.5 and id<40 and name= 'Adrian Ionescu'

The command returns all the records along with all the attributes specified in the WHERE clause.

In this case the main module receives the SELECT command and checks the user's rights. If the user has enough

rights to execute select commands, calls the SELECT Module to search data into the database.

In the implemented version of the server there is accepted only to retrieve information from a single table. There are also not accepted sub-queries.

D. Visual Content-Based Queries

The MMDBMS makes a series of validations, both in the table design phase and after that, when adding values to the database: unique names for databases, tables and, attributes, checking the constraints, etc. The users have the possibility to build content-based visual queries in a simple manner, to the image level. The elements which can be used to build such a query are: like, select, from, where, threshold, maximum number of returned images. The retrieval operations use two functions for computing the similitude between two images: one for texture (Euclidian distance) and the other for color characteristics (histograms intersection). The user has the possibility to choose if he wishes to use both or only one of them. Based on the user's settings, the system will create a modified SQL command, adapted for this type of query.

A special select command is included which can be used for content-based retrieval operations. When the user sends a SELECT_IMAGE command to the system, the DBMS will try to find all the records in the table that contain similar images. This is the most complex operation that has been implemented. The parameters received are: the attributes that should be returned, the table where to search, methods that should be used for computing the similitude, maximum number of returned images, and the image itself. After receiving this command, the server waits to receive the query image from the client. After the image is transferred, it will be saved temporary on the disc using a name that is unique. This is needed in case several clients send the same image simultaneously to the server. After receiving the image, it is processed first and extracted the characteristics. The system can go further only after this step is finish. To the next step it will search the database for similar images using for similarity the specified method [16].

Before being executed, the command is added to the transaction file.

The syntax of the query is:

SelectImage [/<attribute_name1>, <attribute_name2>, ...] FROM <table_name> WHERE <image_column_name> LIKE QueryImage (METHOD: color[,texture] MaxImages <no_of_returned_images>)*

Where:

"attribute_name1" is the attributes that should be returned

"table_name" is the name of the table

"image_column_name" is thename of the column that contains the image itself

"no_of_returned_images" specifies the maximum number of images returned

E.g.:

selectImage name, picture from picture where picture Like QueryImage (method: color maxImages 5)

selectImage * from Patients where age>50 and picture like queryimage

In the first example it is specified that the results are from table "picture". There will be retrieved only the name and picture attributes for the images similar with the query image that is specified by the user. The result will contain only the most similar 5 images.

This type of query can be visually built in two different ways. In the first case the user sends a SELECT ALL query that will retrieve all the records in the table. From these records it selects the image he wishes to be used as a query and then sends a SELECT_IMAGE query with it. The DBMS returns all records containing similar images. The second possibility is when the user has his own image, he calls the Processing Module first and creates the image type value. Then it is sent to the system a SELECT_IMAGE command.

When inserting an image in an Image data type, the Image Processing Module will be automatically called after the image is selected. This will automatically extract the color information and texture (represented by a 12 values vector).

The execution time depends mainly to the number and size of images that will be sent to the client. This is due to the fact that sending the images over the TCP is much slower that processing the images, especially when involves a high number of images.

E. Administration Suport

The MMDBS offer also support for users' management: creating new users, deleting users and managing their access rights.

The user who creates a database will have absolute rights over it. All the other users excepting the administrator will need to receive rights access in order to execute operations over it. There are two kind of rights: general rights (given only by the administrator) and particular rights (for a specified database), given by the administrator or by the database owner.

The general rights refer to:

- Permissions for creating databases
- Permissions for creating other users.

The particular rights that can be given (for each database in part) are:

- Execute select operations
- Execute update/delete operations
- Create Tables
- Modify tables structures
- Delete table/database

The commands are:

1. *create user <user_name> password <password> cd=[0/1] cu=[0/1]*

e.g.: create user cosmin password test cd=1 cu=0

This command will create a new user with the name "cosmin" and the password "test". He will have defined from the beginning the rights to create new databases (cd = 1 for granting this right, or cd = 0 for denying the right) and to create other users (cu = 1 or cu = 0).

For the moment the user doesn't have any right over the existing databases. He cannot execute any commands, as he has no rights granted.

2. *update user rights <nume_user> on <nume_baza_de_date> set ct=1[/0] s=1[/0] u=1[/0] m=1[/0]*

e.g.: update user rights cosmin on person set ct=1 s=1 u=0 m=0

This command is used for granting users rights at the databases level. The rights that can be specified are:

- create table (ct = 1 – granting the right ; ct=0 – denying this right),
- executing select commands(s = 1 – granting the right; s = 0 – denying this right)
- updating information using update/insert (u = 1 – granting the right ; u=0 – denying this right)
- modifying table structure (m = 1 – granting the right ; m=0 – denying this right)

3. *get user rights <user_name> on <database_name>*

e.g.: get user rights cosmin on person

The function is useful to verify the user's rights for a specific database. If only the general rights are it should be specified the "default" database.

4. *get table <table_name> metadata*

e.g.: get table person metadata

The command returns the structure of a table (defined attributes and their data types). It is useful for a client in order to interpret and display correctly the information received from server. Before executing this type of command it should be executed first USE DATABASE command.

5. *set user password <user_name>,<password>*

e.g.: set user password cosmin, test01

The command permits changing users' passwords. Each user can change his password, but he cannot change other's users' password. The system administration is the only one who can do this.

6. *get databases list*

It returns to the user the list with all the databases defined in the system. It is very useful for the client's application in order to know where to search for table lists.

7. *get tables list*

It returns to the user the list with all the tables defined in the databases. It is very useful for the client's application in order to know in which table can search for metadata.

In order to increase the retrieval speed it is used the Clustering module. In this case, first it is searched for the cluster that contains images most similar with the query image. After this cluster is located, the system looks for similar images only inside it. In this way the quantity of data needed to be compared is reduced substantially.

V. CLUSTERING THE IMAGES

In order to increase the speed of the retrieval process, after an image is inserted in the database, it should be assigned to a cluster containing similar images.

For this operation we chose to use the algorithms provided by Weka package [12]. Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

The Expectation-Maximization (EM) algorithm from Weka clustering package EM is a statistical model that makes use of the finite Gaussian mixtures model. The algorithm is similar to the K-means procedure in that a set of parameters are re-computed until a desired convergence value is achieved [14][15].

It needs the input data to be in a custom format called arff. Under these circumstances we have developed an offline Java application that queries the database and creates the input data file called activity.arff. This process is automated and is driven by a property file in which there is specified what data will lay in activity.arff file.

The most important step in this procedure is the attribute selection and the granularity of their nominal values. The number of attributes and their meaning has a crucial importance for the whole process, since irrelevant attributes may degrade classification performance in sense of relevance. On the other hand, the more attributes we have the more time the algorithm will take to produce a result. Domain knowledge and of course common sense, are crucial assets for obtaining relevant results.

For an image we may have a very large number of attributes. Still, in our procedure we used only two sets: color characteristics and texture characteristics. Here is how the arff file looks like:

```
@relation images
@texture
        {v1,v2,v3,v4,v5, .. v12}
@ color
        {v1,v2,v3,v4,v5,…,v166}
@data
Texture1 v1,v2,…
Color1 v1,v2, ..
Texture2 v1,v2,…
Color2 v1,v2, ..
….
```

As it can be seen from the definition of the attributes each of them has a set of five nominal values from which only one may be assigned. The values of the attributes are computed for each of the XXXX images and are set in the @data section of the file. For example, the first line says that the image 1 has value v1 for parameter named color1 and has value v1 for parameter named texture2.

The granularity for the nominal values of the attributes can be also increased. In our study we considered only five possible values but we can consider testing the algorithm with more possible values. This should have great impact on the number of the clusters obtained. The time taken by the algorithm to produce results should also increase.

Running the EM algorithm created XX (e.g. three) clusters. The procedure clustered XX (e.g. 91) instances (34%) in cluster 0, 42 instances (16%) in cluster 1 and 135 instances (50%) in cluster 3. The final step is to check how well the model fits the data by computing the likelihood of a

set of test data given the model. Weka measures goodness-of-fit by the logarithm of the likelihood, or log-likelihood: and the larger this quantity, the better the model fits the data. Instead of using a single test set, it is also possible to compute a cross validation estimate of the log-likelihood. For our instances the value of the log-likelihood is -2.61092 which represent a promising result in the sense that instances (in our case images) may be classified in three disjoint clusters based on their characteristics.

This function of the database is activated when the user sends an INSERT/UPDATE command and after processing the image. The system calls the Clustering Module in order to include the newly inserted image in a cluster. These clusters will be used for retrieval when user sends a SELECT command. The main problem in this case for the Clustering Module is to determine which is the cluster containing similar images. After this cluster is located, only images contained here are computed in order to find images with the highest similarity to the query image.

VI. CLIENT-SERVER COMMUNICATION

The implemented server is a client-server application based on TCP/IP sockets. Each client must have a copy of the client application in order to have access to databases on the server. The client application can be implemented both as a standalone application that can be installed on the client's computer and as an application opened from the internet browser. The second option is recommended because it is not necessary any installation and can be used on any computer with internet access.

The first step is to start the server in order to wait for clients' connections. A condition for the Microsoft Vista operating systems (and future versions based on this architecture) is to execute the server with administrative rights. If not, the server will not be able to receive connections from clients.

The second step is to create a connection between client and server. For this, the client application needs to know the IP address and port where to find the server. The port is specified in the configuration file on the server.

When a client connects to the server, all the communication will be managed on a separated execution thread, independent to all the other connections.

In the third step, the client has to authenticate to the server based on a user and password. The password is defined when the user was added into the system and it is stored in the DEFAULT database. The password is encrypted for safety, on a key based on the user's name. If the user fails to enter the right password for three times, the IP address used to log-in will be blocked for a period of 30 minutes. During this period it will not be accepted any connection from that address.

After the client is authenticated, the server can accept commands. These commands can be only based on text or they can imply sending image files.

The commands that imply only text are:

a) INSERT

If the table where the insert is made does not contain any Image data attribute, the command will need to transfer only text information. The client's command will be received by the central module of the server. It will analyze and recognize that it is only a text-based insert operation and it is not needed to receive anything else from the client (e.g., an image). It will be sent further for execution to the Update module. This module responds with an "OK" message if the insert was successful or with an error message if one of the table's constraints did not permit the insert (e.g., primary key duplication). The message is sent to the "Response" module that returns it to the client.

b) SELECT

If the table used does not contain any image data type attribute, the communication will involve only text. The server receives the command, checks and sees that it is not needed to receive any other information and sends it to the "select" module. This module makes the search and put the result in a special structure that will be returned to the "Response" module. The client receives first the metadata. Only after this it will know how to correctly interpret the following information. On the next step, the server sends the number of records that will be sent to the client. At the end it sends record by record all the information returned by the select.

c) Obtaining databases list, available on the server.

This type of requests will be treated in a similar manner. The server receives the command, checks its type and calls the corresponding module. The response will be a string array.

The commands that might need images transfer are:

a) INSERT

If the table used for insertion has an attribute of image data type, the server will need to exchange text data and image data. The command sent by the client is received first by the main module of the server. For communication with the client it is used a handshaking method: client sends the insert command to the server, the server responds with an OK. It checks and recognizes the insert operation that needs an image. Next, the communication module sends a message to the client to announce that it waits to receive an image. The client responds with the image itself. The server confirms then with an OK receiving the image. If one of them does not receives the confirmation, it will repeat sending the information.

The received image is saved in a temporary folder. The identification of the execution thread will be added to the name of the file in order to ensure that each name is unique.

The "Characteristics extraction" module will process the image and extract the characteristics.

The "Update" module will be called only after the last one finishes processing and all the information needed to make the insertion is available: the text information will be inserted in the table file and the image in the images file.

At the end, the Update Module will return an "OK" message if insertion was successful or an error message if it will be violated one of the table's constraints.

The following figure presents the communication with the client:

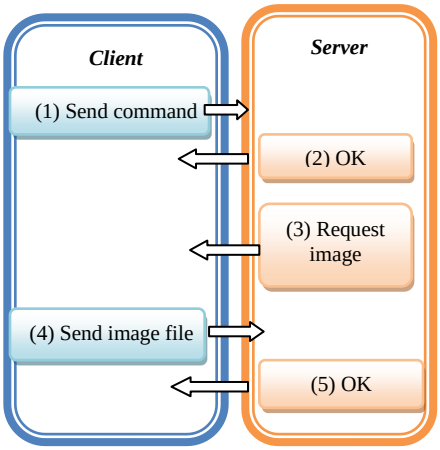

Figure 2. Client-server communication.

The execution time of this command depends on two components: the time needed to send the image to the server (link speed) and the time needed to process the image (processor speed). These components depend mainly on the image size.

VII. SERVER FILES STRUCTURE

Because the images are stored directly in the database it could be a problem the files' size on the disc. The size of the images file can grow very much, reaching easily several hundred MB, or even few GB. This is the main reason why it is recommended to use the NTFS file system [2].

The Database File Manager Module is the module that effectively read and writes from/to files. It uses the pagination reading and writing, all information being written in pages of information. A single file can contain several types of information that are written in different type of pages. In order to identify the type of each page, all the pages are indexed in an index file.

The server uses the following types of files:

- Users.ssc – it is the file where there are stored all users created on the server. For each user it is stored two set of rights: general rights and rights detailed for each database. Each of these types of information is stored in a separate page.
- Index_users.ssc – it is the file where the pages used to store users' information are indexed. There can be two types of pages: pages used to store users information and pages used to store users' rights.
- Transaction_log.ssc – it is a file created for each database. It stores the transaction log history. It can be used for database recovery in case of a system error.
- <table_name>.ssc – it is the table file. This file contains two types of data: metadata information and data information.
- Page_Index_<table_name>.ssc – it is the file where all the pages from table file are indexed for a faster retrieval. The pages can be used to store either metadata, either useful information.
- Images.ssc – on the table page there is stored only the characteristics of the image. The image itself it is stored on a special file called images.ssc.
- Index_<table_name>_<key>.ssc – this type of file is optionally and is created by the user if he wishes to index the information in table by one of its columns (key).

The server creates a new folder for each database. The name of the folder is considered to be the name of the database. The user who created the database will have absolute rights over it. All the other users excepting the administrator will need to receive rights access in order to execute operations over it. There are two kind of rights: general rights (given only by the administrator) and particular rights (for a specified database), given by the administrator or by the database owner.

B. Reading and writing information

When the server receives a command for an update, insert or select, it has to store the operation in the transaction log file before doing any other operation. The record will include information about the user who sent the command and about the operation that should be executed. Only after this process is finished, the system goes further to the next step and tries to read or write from files. In order to know how to interpret the information that it will read, it has to find the structure of the table first. It will search in the page_index_table_name file the page containing table metadata and read it. Now, the system can read and "understand" correctly the information from table and can separate correctly each attribute's data. There are two possible cases for reading data. In the first case, the user has already created an index for the specified key using a Create Index command. In this case a file called Index_table_name_key should exist. It will be searched for the page containing the key and read into memory from the table file. Only after that it can be read record by record to find the right information. In the second case, the information is not indexed. In this case each page existing in page_index_table_name file is read in the memory and then read record by record from the memory to find the right information.

The algorithm used for reading is presented next:

```
Procedure Read_info (key)
 Find in Page_Index_table_name metadata page
 Read corresponding page from memory
 If exists index (key)
   Find key in table      Index_table_name_key
      If information found then
                Read corresponding page to memory
                Find key in the page
      Else
                Information not found
 Else
      For each page of information from
      Page_Index_table_name
                Read corresponding page from memory
                Find key in the page
```

When the system tries to write information in the file it is searched in the page_index_table_name file the last page containing information. There are two cases: there is enough free space on the page to write the current data (in this case it is written after the last record in the page and then updated the information from page_index_table_name file) or there is not enough space to write in the current page. In this case it is created a new page where the data is written and then writes the details of this page to the page_index_table_name file. If the written information contains a key that is indexed it is called the function update index. The algorithm is:

```
Procedure Write_info (info)
 Find in the page_index_table_name last page with info
 Read corresponding page to memory
 If firstFreePositionInPage + size (info) < pageSize
      Write info to page
      Update page information in Page_Index_table_
name
 Else
      Create a new page
      Write info to page
      Add page information in Page_Index_ table_name
 If exists index (info)
      Update index file.
```

If the user sent an update command the system will locate the existing information, update data and then write page back to the disk. For locating the data that should be updated it is used the Read_Info procedure described above. After the information is updated it is possible that the size of the record to be changed. In this case the information from page will be reorganized. It is deleted the record from its current location and added to the end. If the page free space is not big enough to include the current information the page is written back to disk without the modified information and the information will be added to a new page.

The algorithm is described next:

```
Procedure Update_info (key
 Read_info (key)
 Update info
 Delete_info(page, info)
 If firstFreePositionInPage + size (info) < pageSize
      Write info to page
      Update page information in Page_Index_table_
name
 Else
      Create a new page
      Write info to page
      Add page information in Page_Index_ table_name
 If exists index (info)
      Update index file.
```

VIII. CONCLUSIONS

The paper presents the organization of an implemented multimedia, relational, database multimedia server kernel that included a new data type, called IMAGE.

It is created for managing and querying medium sized personal digital collections that contain both alphanumerical information and digital images (for examples the ones used in private medical consulting rooms). The software tool allows creating and deleting databases, creating and deleting tables in databases, updating data in tables and querying. The user can use several types of data: integer, char, double and image. There are also implemented the two constraints used in the relational model: primary key and referential integrity.

The software tool can execute both simple text-based queries using one or several criteria connected with logical operators (and, or) and content-based visual queries at image level, taking into consideration the color and texture characteristics. These characteristics are automatically extracted when the images are inserted in the database.

It is presented the algorithms used for reading/writing the information into the database, along with the files used by the server.

Storing the images inside the database can easily lead to very large files that can go beyond several GB. That is why the server administrator has to take care of the available disc space. It is also recommended that the server to be installed on a NTFS file system that has almost no size limit for files, comparing to the FAT file system that accepts files up to 4 GB in size.

This software can be extended in the following directions:

- Adding new traditional and multimedia data types (for example video data type or DICOM type - the main area where this multimedia DBMS is used it is the medical domain. The DICOM type of data is used for storing alphanumerical information along with images existing in a standard DICOM file that is provided by a medical device)
- Studying and implementing indexing algorithms for data inserted in the tables.

The speed of the image retrieval module is increased by using a clustering module that will group the images in clusters. The similarity between a query image and the images in the database will be computed only taking into consideration the images from the cluster with the best similarity.

REFERENCES

[1] C. Stoica Spahiu, C. Mihaescu, L. Stanescu, D.D. Burdescu, and M. Brezovan, "Database Kernel for Image Retrieval",. Proceedings of The First International Conference on Advances in Multimedia (MMEDIA 2009), Colmar – France, pp. 169-173, 2009

[2] C. Stoica Spahiu, Liana Stanescu, D.D. Burdescu, and M. Brezovan, "File Storage for a Multimedia Database Server for Image Retrieval", Proceedings of The Fourth International Multi-Conference on Computing in the Global Information Technology (ICCGI 2009), Cannes/ La Bocca – France, France, pp.35-40, 2009

[3] SQL Server 2008 Books Online, January 2009, http://msdn.microsoft.com/en-us/library/ms187993.aspx

[4] MySQL 5.0 Reference Manual, 2009 http://dev.mysql.com/doc/refman/5.0/en/blob.html

[5] A. Chigrik, SQL Server 2000 vs Oracle 9i, 2007 www.mssqlcity.com/Articles/Compare/sql_server_vs_oracle.htm

[6] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," Proc. of the 20th VLDB Conference, pp. 487-499, Santiago, Chile, 1994.

[7] M. Kratochvil, The Move to Store Images In the Database, 2005 http://www.oracle.com/technology/products/intermedia/pdf/why_images_in_database.pdf

[8] Oracle® Database SQL Reference 10g Release 2 (10.2) http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/sql_elements001.htm

[9] New interMedia Features in Oracle10g http://www.oracle.com/technology/products/intermedia/htdocs/imedia _new_features_in_10g.html

[10] M. Ester, H.P.Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", Proc. KDD'96, Portland, OR, pp.226-231,1996.

[11] R. Sibson, "SLINK: An Optimally Efficient Algorithm for the Single-link Cluster Method", The Computer Journal, 16(1): 30-34, 1973.

[12] www.cs.waikato.ac.nz/ml/weka

[13] L. Stanescu, D.D. Burdescu, M. Brezovan, C. Stoica Spahiu, and A. Ion, "A New Software Tool For Managing and Querying the Personal Medical Digital Imagery", Proceedings of the International Conference on Health Informatics, Porto – Portugal, pp. 199-204, 2009

[14] http://grb.mnsu.edu/grbts/doc/manual/Expectation_Maximization _EM.html

[15] James Foulds. Learning instance weights in multi-instance learning. Master's thesis, Department of Computer Science, University of Waikato, 2008 http://adt.waikato.ac.nz/uploads/approved/adt-uow20080308.150730/public/02whole.pdf

[16] L. Stanescu, M.C. Mihaescu, D.D. Burdescu, E. Georgescu, and L. Florea, "An Improved Platform for Medical E-Learning", Lecture Notes in Computer Science 4823, Springer, pp.392-403, 2008.